Finite LTL Synthesis with Environment Assumptions and Quality Measures

Alberto Camacho, Meghyn Bienvenu, Sheila Mcilraith

Finite LTL Synthesis with Environment Assumptions and Quality Measures

Alberto Camacho
Department of Computer Science
University of Toronto, Canada
acamacho@cs.toronto.edu

Meghyn Bienvenu
CNRS, Univ. Montpellier, Inria
Montpellier, France
meghyn@lirmm.fr

Sheila A. McIlraith
Department of Computer Science
University of Toronto, Canada
sheila@cs.toronto.edu

Abstract

In this paper, we investigate the problem of synthesizing strategies for linear temporal logic (LTL) specifications that are interpreted over finite traces – a problem that is central to the automated construction of controllers, robot programs, and business processes. We study a natural variant of the finite LTL synthesis problem in which strategy guarantees are predicated on specified environment behavior. We further explore a quantitative extension of LTL that supports specification of quality measures, utilizing it to synthesize high-quality strategies. We propose new notions of optimality and associated algorithms that yield strategies that best satisfy specified quality measures. Our algorithms utilize an automata-game approach, positioning them well for future implementation via existing state-of-the-art techniques.

1 Introduction

The problem of automatically synthesizing digital circuits from logical specifications was first proposed by Church (1957). In 1989, Pnueli and Rosner examined the problem of synthesizing strategies for reactive systems, proposing Linear Temporal Logic (LTL) (Pnueli 1977) as the specification language. In a nutshell, LTL is used to express temporally extended properties of infinite state sequences (called traces), and the aim of LTL synthesis is to produce a winning strategy, i.e. a function that assigns values to the state variables under the control of the system at every time step, in such a way that the induced infinite trace is guaranteed to satisfy the given LTL formula, no matter how the environment sets the remaining state variables.

In 2015, De Giacomo and Vardi introduced the problem of *LTL_f synthesis* in which the specification is described in a variant of LTL interpreted over finite traces (De Giacomo and Vardi 2013). Finite interpretations of LTL have long been exploited to specify temporally extended goals and preferences in AI automated planning (e.g., (Bacchus and Kabanza 2000; Baier, Bacchus, and McIlraith 2009)). In contrast to LTL synthesis, which produces programs that run in perpetuity, $\mathsf{LTL_f}$ synthesis is concerned with the generation of terminating programs. Two natural and important application domains are automated synthesis of business processes, including web services; and automated synthesis of robot controllers, in cases where program termination is desired.

Despite recent work on $\mathsf{LTL_f}$ synthesis, there is little written on the nature and form of the $\mathsf{LTL_f}$ specifications and how this relates to the successful and nontrivial realization of strategies for such specifications. $\mathsf{LTL_f}$ synthesis is conceived as a game between the environment and an agent. The logical specification that defines the problem must not only define the desired behavior that execution of the strategy should manifest – what we might loosely think of as the *objective* of the strategy, but must also define the context, including any *assumptions* about the environment's behavior upon which realization of the objective is predicated. As we show in this work, if assumptions about environment behavior are not appropriately taken into account, specifications can either be impossible to realize or can be realized trivially by allowing the agent to violate assumptions upon which guaranteed realization of the objective is predicated.

We further examine the problem of how to construct specifications where the realization of an objective comes with a quality measure, and where strategies provide guarantees with respect to these measures. The addition of quality measures is practically motivated. In some instances we may have an objective that can be realized in a variety of ways of differing quality (e.g., my automated travel assistant may find a myriad of ways for me to get to KR2018 – some more preferable than others!). Similarly, we may have multiple objectives that are mutually unachievable and we may wish to associate a quality measure with their individual realization (e.g., I'd like my home robot to do the laundry, wash dishes, and cook dinner before its battery dies, but dinner is most critical, followed by dishes).

In this paper we explore finite LTL synthesis with environment assumptions and quality guarantees. In doing so, we uncover important observations regarding the form and nature of $\mathsf{LTL_f}$ synthesis specifications, how resulting strategies are computed, and the nature of the guarantees we can provide regarding the resulting strategies. In Section 3 we examine the problem of $\mathsf{LTL_f}$ synthesis with environment assumptions, introducing the notion of *constrained* $\mathsf{LTL_f}$ *synthesis* in Section 4. In Section 5, we propose algorithms for constrained $\mathsf{LTL_f}$ synthesis, including a reduction to Deterministic Büchi Automata games for the fragment of environment constraints that are conjunctions of safe and co-safe LTL formulae. In Section 6, we examine the problem of augmenting constrained $\mathsf{LTL_f}$ synthesis with quality measures. We adopt a specification language, $\mathsf{LTL_f}[\mathcal{F}]$, proposed by (Almagor et al. 2017) and define a new notion of optimal strategies. In Section 7, we provide algorithms for computing high-quality strategies for constrained $\mathsf{LTL_f}$ synthesis. Section 8 summarizes our technical contributions. Some proofs are deferred to the appendix. This paper (without the appendix proofs) will appear in the *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning* (KR 2018).

2 Preliminaries

We recall the syntax and semantics of linear temporal logic for both infinite and finite traces, as well as the basics of finite state automata and the link between LTL and automata.

2.1 Linear Temporal Logic (LTL)

Given a set $\mathcal{P}$ of propositional variables, LTL formulae are defined as follows:

$$\varphi \coloneqq \top \mid \bot \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2 \mid \varphi_1 \mathsf{R} \varphi_2$$

where $p \in \mathcal{P}$. Here $\neg$, $\wedge$, and $\vee$ are the usual Boolean connectives, and $\bigcirc$ (next), U (until), and R (release) are temporal operators. The formula $\bigcirc\varphi$ states that φ must hold in the next timepoint, $\varphi_1 \mathsf{U} \varphi_2$ stipulates that φ_1 must hold until φ_2 becomes true, and $\varphi_1 \mathsf{R} \varphi_2$ expresses that φ_2 remain true until and including the point in which φ_1 is made true (or forever if φ_2 never becomes true). For concision, we do not include logical implication ($\rightarrow$), eventually ($\Diamond$, 'sometime in the future') and always ($\Box$, 'at every point in the future') in the core syntax, but instead view them as abbreviations: $\alpha \rightarrow \beta \coloneqq \neg\alpha \vee \beta$, $\Diamond\varphi \coloneqq \top \mathsf{U} \varphi$ and and $\Box\varphi \coloneqq \bot \mathsf{R} \varphi$.

LTL formulae are traditionally interpreted over *infinite traces* π, i.e., infinite words over the alphabet $2^{\mathcal{P}}$. Intuitively, an infinite trace π describes an infinite sequence of (time)steps, with the i-th symbol in π, written $\pi(i)$, specifying the propositional symbols that hold at step i. We use $\pi \sqsubseteq \pi'$ to indicate that π is a prefix of π'. We define what it means for an infinite trace π to *satisfy an* LTL *formula* φ *at step* i, denoted $\pi \models_i \varphi$:

- $\pi \models_i \top$, $\pi \not\models_i \bot$, and $\pi \models_i p$ iff $p \in \pi(i)$, for each $p \in \mathcal{P}$;
- $\pi \models_i \neg\varphi$ iff $\pi \not\models_i \varphi$;
- $\pi \models_i \varphi_1 \wedge \varphi_2$ iff $\pi \models_i \varphi_1$ and $\pi \models_i \varphi_2$;
- $\pi \models_i \varphi_1 \vee \varphi_2$ iff $\pi \models_i \varphi_1$ or $\pi \models_i \varphi_2$;
- $\pi \models_i \bigcirc\varphi$ iff $\pi \models_{i+1} \varphi$;
- $\pi \models_i \varphi_1 \mathsf{U} \varphi_2$ iff there exists $j \geq i$ such that $\pi \models_j \varphi_2$, and for each $i \leq k < j$, $\pi \models_k \varphi_1$;
- $\pi \models_i \varphi_1 \mathsf{R} \varphi_2$ iff for all $j \geq i$ either $\pi \models_j \varphi_2$ or there exists $i \leq k < j$ such that $\pi \models_k \varphi_1$.

A formula φ is *satisfied in* π, written $\pi \models \varphi$, if $\pi \models_1 \varphi$. Two formulas φ and ψ are *equivalent* if $\pi \models \varphi$ iff $\pi \models \psi$ for all traces π. Observe that, in addition to the usual Boolean equivalences, we have the following: $\varphi_1 \mathsf{U} \varphi_2 \equiv \neg(\neg\varphi_1 \mathsf{R} \neg\varphi_2)$ and $\neg\bigcirc\varphi \equiv \bigcirc\neg\varphi$.

We consider two well-known syntactic fragments of LTL. The *safe fragment* is defined as follows (Sistla 1994):

$$\varphi \coloneqq \top \mid \bot \mid p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{R} \varphi_2$$

The complementary *co-safe fragment* is similarly defined, using U in place of R. It is known that if φ is a safe formula and $\pi \not\models \varphi$, then there is a finite *bad prefix* $\pi_b \sqsubseteq \pi$ such that $\pi' \not\models \varphi$ for every infinite trace π' with $\pi_b \sqsubseteq \pi'$. Similarly, if φ is a co-safe formula and $\pi \models \varphi$, then there exists a finite *good prefix* $\pi_g \sqsubseteq \pi$ such that $\pi' \models \varphi$ for every infinite trace π' with $\pi_g \sqsubseteq \pi'$. This means that violation of safe formulae and satisfaction of co-safe formulae can be shown by exhibiting a suitable finite prefix (Kupferman and Vardi 2001).

In this paper, our main focus will be on a more recently studied *finite version of* LTL, denoted $\mathsf{LTL_f}$ (De Giacomo and Vardi 2013), in which formulae are interpreted over *finite traces* (finite words over $2^{\mathcal{P}}$). We will reuse the notation $\pi(i)$ (i-th symbol) and introduce the notation $|\pi|$ for the length of π. $\mathsf{LTL_f}$ has precisely the same syntax as LTL and the same semantics for the propositional constructs, but it differs in its interpretation of the temporal operators:

- $\pi \models_i \bigcirc\varphi$ iff $|\pi| > i$ and $\pi \models_{i+1} \varphi$;
- $\pi \models_i \varphi_1 \mathsf{U} \varphi_2$ iff there exists $i \leq j \leq |\pi|$ such that $\pi \models_j \varphi_2$, and $\pi \models_k \varphi_1$, for each $i \leq k < j$;
- $\pi \models_i \varphi_1 \mathsf{R} \varphi_2$ iff for all $i \leq j \leq |\pi|$ either $\pi \models_j \varphi_2$ or there exists $i \leq k < j$ such that $\pi \models_k \varphi_1$

We introduce the *weak next* operator ($\bullet$) as an abbreviation: $\bullet\varphi \coloneqq \bigcirc\varphi \vee \neg\bigcirc\top$. Thus, $\bullet\varphi$ holds if φ holds in the next time step or we have reached the end of the trace. Over finite traces, $\neg\bigcirc\varphi \not\equiv \bigcirc\neg\varphi$, but we do have $\neg\bullet\varphi \equiv \bigcirc\neg\varphi$.

As before, we say that φ is satisfied in π, written $\pi \models \varphi$, if $\pi \models_1 \varphi$. Note that we can unambiguously use the same notation for LTL and $\mathsf{LTL_f}$ so long as we specify whether the considered trace is finite or infinite.

2.2 Finite State Automata

We recall that a *non-deterministic finite-state automaton (NFA)* is a tuple $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, F \rangle$, where Σ is a finite alphabet of *input symbols*, Q is a finite set of *states*, $Q_0 \subseteq Q$ is a set of *initial states*, $F \subseteq Q$ is a set of *accepting states*, and $\delta : Q \times \Sigma \rightarrow 2^Q$ is the *transition* function. NFAs are evaluated on *finite* words, i.e. elements of Σ^*. A *run* of $\mathcal{A}$ on a word $w = \theta_1 \cdots \theta_n$ is a sequence $q_0 \cdots q_n$ of states, such that $q_0 \in Q_0$, and $q_{i+1} \in \delta(q_i, \theta_{i+1})$ for all $0 \leq i < n$. A run $q_0 \cdots q_n$ is *accepting* if $q_n \in F$, and $\mathcal{A}$ *accepts* w if some run of $\mathcal{A}$ on w is accepting. The *language* of an automaton $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of words accepted by $\mathcal{A}$.

Deterministic finite-state automata (DFAs) are NFAs in which $|Q_0| = 1$ and $|\delta(q, \theta)| = 1$ for all $(q, \sigma) \in Q \times \Sigma$. When $\mathcal{A}$ is a DFA, we write $\delta : Q \times \Sigma \rightarrow Q$ and $q' = \delta(q, \theta)$ in place of $q' \in \delta(q, \theta)$, and when $Q_0 = \{q_0\}$, we will simply write q_0 (without the set notation). For every NFA $\mathcal{A}$, there exists a DFA that accepts the same language as $\mathcal{A}$ and whose size is at most single exponential in the size of $\mathcal{A}$. The *powerset construction* is a well-known technique to *determinize* NFAs (Rabin and Scott 1959).

Non-deterministic Büchi automata (NBA) are defined like NFAs but evaluated on *infinite* words, that is, elements of Σ^ω. A *run* of $\mathcal{A}$ on an infinite word $w = \theta_1\theta_2 \cdots$ is a sequence $\rho = q_0 q_1 q_2 \cdots$ of states, such that $q_0 \in Q_0$, and $q_{i+1} \in \delta(q_i, \theta_{i+1})$ for every $i \geq 0$. A run ρ is *accepting* if $\mathsf{inf}(\rho) \cap F \neq \emptyset$, where $\mathsf{inf}(\rho)$ is the set of states that appear infinitely often in ρ. We say that an NBA $\mathcal{A}$ *accepts* w if some run of $\mathcal{A}$ on w is accepting. Analogous definitions apply to *deterministic* Büchi automata (DBAs).

We will also consider *deterministic finite-state transducers* (also called Mealy machines, later abbreviated to 'transducers'), given by tuples $\mathcal{T} = \langle \Sigma, \Omega, Q, \delta, \omega, q_0 \rangle$, where Σ and Ω are respectively the *input and output alphabets*, Q is the set of states, $\delta : Q \times \Sigma \to Q$ is the transition function, $\omega : Q \times \Sigma \to \Omega$ is the *output function*, and q_0 the initial state. The run of $\mathcal{T}$ on $w = \theta_1\theta_2 \ldots \in \Sigma^\omega$ is an infinite sequence of states $q_0q_1q_2\ldots$ with $q_{i+1} \in \delta(q_i, \theta_{i+1})$ for every $i \geq 0$, and the *output sequence* of $\mathcal{T}$ on w is $\omega(q_0, \theta_1)\omega(q_1, \theta_2)\ldots$.

Given an $\mathsf{LTL_f}$ formula φ, one can construct an NFA that accepts precisely those finite traces π with $\pi \models \varphi$ (e.g. (De Giacomo and Vardi 2015)). For every safe formula φ_s (resp. co-safe formula φ_c), one can construct an NFA that accepts all bad prefixes of φ_s (resp. good prefixes of φ_c) (Kupferman and Vardi 2001). In these constructions, the NFAs are worst case single exponential in the size of the formula. By determinizing these NFAs, we can obtain DFAs of double-exponential size that recognize the same languages.

3 LTL and $\mathsf{LTL_f}$ Synthesis

To set the stage for our work, we recall the definition of LTL synthesis in the infinite and finite trace settings and the relationship between planning and synthesis.

3.1 LTL Synthesis

An LTL *specification* is a tuple $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ where φ is an LTL formula over *uncontrollable* variables $\mathcal{X}$ and *controllable* variables $\mathcal{Y}$. A *strategy* is a function $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$. The *infinite trace induced by* $\mathbf{X} = \{X_i\}_{i\geq 1} \in (2^{\mathcal{X}})^\omega$ *and* σ is

$$\pi[\sigma, \mathbf{X}] = (X_1 \cup \sigma(X_1))\,(X_2 \cup \sigma(X_1X_2))\ldots$$

The set of all infinite traces induced by σ is denoted $\mathsf{traces}(\sigma) = \{\pi[\sigma, \mathbf{X}] \mid \mathbf{X} \in (2^{\mathcal{X}})^\omega\}$. The *realizability* problem $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ consists in determining whether there exists a *winning strategy*, i.e., a strategy σ such that $\pi \models \varphi$ for every $\pi \in \mathsf{traces}(\sigma)$. The *synthesis* problem is to compute such a winning strategy when one exists.

LTL synthesis can be viewed as a 2-player game between the environment ($\mathcal{X}$) and the agent ($\mathcal{Y}$). In each turn, the environment makes a move by selecting $X_i \subseteq \mathcal{X}$, and the agent replies by selecting $Y_i \subseteq \mathcal{Y}$. The aim is to find a strategy σ for the agent that guarantees the resulting trace satisfies φ.

3.2 Finite LTL Synthesis

We now recall $\mathsf{LTL_f}$ realizability and synthesis, where the specification formula is interpreted on finite traces. An $\mathsf{LTL_f}$ *specification* is a tuple $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$, where φ is an $\mathsf{LTL_f}$ formula over *uncontrollable* variables $\mathcal{X}$ and *controllable* variables $\mathcal{Y}$. A *strategy* is a function $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y} \cup \{\mathsf{end}\}}$ such that for each infinite sequence $\mathbf{X} = \{X_i\}_{i\geq 1} \in (2^{\mathcal{X}})^\omega$ of subsets of $\mathcal{X}$, there is exactly one integer $n_{\sigma,\mathbf{X}} \geq 1$ with $\mathsf{end} \in \sigma(X_1 \cdots X_{n_{\sigma,\mathbf{X}}})$. The induced infinite trace $\pi[\sigma, \mathbf{X}]$ is defined as before, and the *finite trace induced by* $\mathbf{X}$ *and* σ is

$$\pi^{\mathsf{f}}[\sigma, \mathbf{X}] = (X_1 \cup \sigma(X_1))\ldots\left(X_{n_{\sigma,\mathbf{X}}} \cup \sigma(X_1 \cdots X_{n_{\sigma,\mathbf{X}}})\right)$$

but with end removed from $\sigma(X_1 \cdots X_{n_{\sigma,\mathbf{X}}})$. The set of all *finite traces induced by* σ is denoted $\mathsf{traces}^{\mathsf{f}}(\sigma) = \{\pi^{\mathsf{f}}[\sigma, \mathbf{X}] \mid \mathbf{X} \in (2^{\mathcal{X}})^\omega\}$. A finite trace π is *compatible with* σ if $\pi \sqsubseteq \pi'$ for some $\pi' \in \mathsf{traces}^{\mathsf{f}}(\sigma)$, with $\mathsf{ptraces}(\sigma)$ ('p' for 'partial') the set of all such traces. We call σ a winning strategy for an $\mathsf{LTL_f}$ specification $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ if $\pi \models \varphi$ for every $\pi \in \mathsf{traces}^{\mathsf{f}}(\sigma)$. The realizability and synthesis problems for $\mathsf{LTL_f}$ are then defined in the same way as for LTL.

Comparison with prior formulations Prior work on $\mathsf{LTL_f}$ synthesis defined strategies as functions $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ that do not explicitly indicate the end of the trace (De Giacomo and Vardi 2015; Zhu et al. 2017; Camacho et al. 2018a). In these works, a strategy σ is winning iff for each $\pi \in \mathsf{traces}(\sigma)$ there exists some finite prefix $\pi' \sqsubseteq \pi$ such that $\pi' \models \varphi$. Note that in general, multiple prefixes π' that satisfy φ may exist. We believe that it is cleaner mathematically to be precise about which trace is produced, and it will substantially simplify our technical developments. The two definitions give rise to the same notion of realizability, and existing results and algorithms for $\mathsf{LTL_f}$ synthesis transfer to our slightly different setting.

3.3 Planning as $\mathsf{LTL_f}$ Synthesis

It has been observed that different forms of automated planning can be recast as $\mathsf{LTL_f}$ synthesis (see e.g. (De Giacomo and Vardi 2015; D'Ippolito, Rodríguez, and Sardiña 2018; Camacho et al. 2018b)). We recall that planning problems are specified in terms of a set of *fluents* (i.e., atomic facts whose value may change over time), a set of *actions* which can change the state of the world, an *action theory* whose axioms give the *preconditions* and *effects* of the actions (i.e., which fluents must hold for an action to be executable, and how do the fluents change as a result of performing an action), a description of the *initial state*, and a *goal*. In classical planning, actions are deterministic (i.e. there is a unique state resulting from performing an action in a given state), and the aim is to produce a sequence of actions leading from the initial state to a goal state. In *fully observable non-deterministic* (FOND) planning, actions have non-deterministic effects, meaning that there may be multiple possible states that result from performing a given action in a given state (with the effect axioms determining which states are possible results). Strong solutions are *policies* (i.e., functions that map states into actions) that guarantee eventual achievement of the goal.

We briefly describe how FOND planning can be reduced to $\mathsf{LTL_f}$ synthesis,[1] as the reduction crucially relies on the use of environment assumptions. We will use the set $\mathcal{F}$ of fluents as the uncontrollable variables, and the set of actions $\mathcal{A}$ for the controllable variables. The high-level structure of the $\mathsf{LTL_f}$ specification formula is: $\Phi = (\Psi_{init} \land \Psi_{eff}) \to (\Psi_{pre} \land \Psi_{goal})$ Intuitively, Φ states that under the assumption that the environment sets the fluents in accordance with the initial state and effect axioms (captured by Ψ_{init} and Ψ_{eff}), the agent can choose a single action per turn (Ψ_{one}) in such a way that the preconditions are obeyed (Ψ_{pre}) and the goal is achieved (Ψ_{goal}). We set $\Psi_{goal} = \Diamond(\gamma \land \neg\bigcirc\top)$,

[1] Our high-level presentation combines elements of the reductions in (De Giacomo and Vardi 2015; Camacho et al. 2018a). Its purpose is to illustrate the general form and components of an $\mathsf{LTL_f}$ encoding of planning (not to provide the most efficient encoding).

where γ is a propositional formula over $\mathcal{F}$ describing goal states. The formula $\Psi_{one} = \Box\psi_{one}$ with $\psi_{one} = (\bigcirc\top \leftrightarrow \bigvee_{a\in\mathcal{A}} a) \wedge \bigwedge_{a,a'\in\mathcal{A},a\neq a'}(\neg a \vee \neg a')$ enforces that a single action is performed at each step. The formula Ψ_{pre} can be defined as $\Box \bigwedge_{a\in\mathcal{A}}(a \rightarrow \rho_a)$, where ρ_a is a propositional formula over $\mathcal{F}$ (typically, a conjunction of literals) that gives the preconditions of a. The formula Ψ_{init} will simply be the conjunction of literals over $\mathcal{F}$ corresponding to the initial state. Finally, Ψ_{eff} will be a conjunction of formulae of the form

$$\Box\left((\kappa \wedge a \wedge \rho_a \wedge \psi_{one}) \rightarrow \bullet\beta\right) \tag{1}$$

where $a \in \mathcal{A}$, and κ and β are propositional formulas over $\mathcal{F}$. Intuitively, the latter formula states that if the current state verifies κ and action a is correctly performed by the agent (i.e. the preconditions are met and no other action is simultaneously performed) then the next state must satisfy β. We discuss later why it is important to include $\rho_a \wedge \psi_{one}$.

3.4 Illustrative Example

We now give a concrete example of an $\mathsf{LTL_f}$ synthesis problem, which illustrates the importance of environment assumptions. Consider synthesizing a high-level control strategy for your Roomba-style robot vacuum cleaner. You want the robot to clean the living room (LR) and bedroom (BR) when they are dirty, but you don't want it to vacuum a room while your cat is there (the robot scares her). We now describe how this problem can be formalized as $\mathsf{LTL_f}$ synthesis.

Taking inspiration from the encoding of planning, we will use $\{clean(z), catIn(z) \mid z \in \{LR, BR\}\}$ (the fluents[2] in our scenario) as the set of uncontrollable variables, and take the robot's actions $\{vac(BR), vac(LR)\}$ as the controllable variables. As was the case for planning, it is natural to conceive of the specification as having the form of an implication $\Psi^{cat}_{env} \rightarrow \Psi^{cat}_{robot}$, with Ψ^{cat}_{env} describing the rules governing the environment's behavior and Ψ^{cat}_{robot} the desired behavior of the robot. We define Ψ^{cat}_{robot} as the conjunction of:

- for $z \in \{LR, BR\}$, the formula $\Box(vac(z) \rightarrow \rho_{vac(z)})$, with $\rho_{vac(z)} = \neg clean(z) \wedge \neg catIn(z)$ the precondition of $vac(z)$ (we can only vacuum dirty cat-free rooms);
- $\Box(\neg vac(LR) \vee \neg vac(BR))$ (we cannot vacuum in two places at once);
- $\Diamond(clean(LR) \wedge clean(BR))$ (our goal: both rooms clean).

We let $\varphi_{vac(z)} = vac(z) \wedge \rho_{vac(z)} \wedge \neg vac(z')$ (with z' the other room) encode a correct execution of $vac(z)$, and let Ψ^{cat}_{env} be a conjunction of the following:

- for $z \in \{LR, BR\}$: $\Box\,(clean(z) \vee \varphi_{vac(z)} \rightarrow \bullet clean(z))$ (if room z is currently clean, or if the robot correctly performs action $vac(z)$, then room z is clean in the next state[3])
- for $z \in \{LR, BR\}$: $\Box\,(\neg clean(z) \wedge \neg vac(z) \rightarrow \bullet\neg clean(z))$ (a room can only become clean if it is vacuumed);
- $\Box(\neg catIn(LR) \vee \neg catIn(BR))$ and $\Box(catIn(LR) \vee catIn(BR))$ (the cat is in exactly one of the rooms).

As the reader may have noticed, while the assumptions in Ψ^{cat}_{env} are necessary, they are not sufficient to ensure realizability, as the cat may stay forever in a dirty room. If we further assume that the cat eventually leaves each of the rooms ($\varphi_{leaves} = \Diamond\neg catIn(BR) \wedge \Diamond\neg catIn(LR)$), there is an obvious solution: vacuum a cat-free room, and then simply wait until the other room is cat-free and then vacuum it. However, rather unexpectedly, adding φ_{leaves} to Ψ^{cat}_{env} makes the specification $\Psi^{cat}_{env} \rightarrow \Psi^{cat}_{robot}$ realizable in a trivial and unintended way: by ending execution in the first move, $\neg\Psi^{cat}_{env}$ trivially holds in the resulting length-1 trace π. Indeed, there are three possibilities: (i) $\pi \models catIn(BR)$ (so $\pi \not\models \Diamond\neg catIn(BR)$), (ii) $\pi \models catIn(LR)$ (so $\pi \not\models \Diamond\neg catIn(LR)$), or (iii) $\pi \models \neg catIn(LR) \wedge \neg catIn(BR)$ (so $\pi \not\models (\Box(catIn(LR) \vee catIn(BR)))$). Clearly, this length-1 strategy is not the strategy that we wanted to synthesize. In Section 4, we propose a new framework for handling environment assumptions which avoids the generation of such trivial strategies and makes it possible to find the desired strategies.

[2]We use notation reminiscent of first-order logic to enhance readability, but the variables (e.g. $clean(LR)$) are propositional.

[3]For simplicity, we assume once a room is clean, it stays clean.

4 Constrained $\mathsf{LTL_f}$ Synthesis

To the aim of properly handling environment assumptions, we introduce a generalization of $\mathsf{LTL_f}$ synthesis, in which the assumptions are separated from the rest of the specification formula and given a different interpretation. Essentially, the idea is that the environment is allowed to satisfy the assumption over the whole infinite trace, rather than on the finite prefix chosen by the agent. This can be accomplished using LTL semantics for the environment assumption, but keeping $\mathsf{LTL_f}$ semantics for the formula describing the objective.

Formally, a *constrained* $\mathsf{LTL_f}$ *specification* is a tuple $\langle\mathcal{X}, \mathcal{Y}, \alpha, \varphi\rangle$, where $\mathcal{X}$ and $\mathcal{Y}$ are the uncontrollable and controllable variables, φ is an $\mathsf{LTL_f}$ formula over $\mathcal{X} \cup \mathcal{Y}$, and α is an LTL formula over $\mathcal{X} \cup \mathcal{Y}$. Here φ describes the desired agent behavior when the environment behaves so as to satisfy α. We will henceforth call φ the *objective*, and will refer to α as the (environment) *assumption* or *constraint* (as it acts to constrain the allowed environment behaviors).

A *strategy for* $\langle\mathcal{X}, \mathcal{Y}, \alpha, \varphi\rangle$ is a function $\sigma : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}\cup\{\mathsf{end}\}}$ such that for each infinite sequence $\mathbf{X} = \{X_i\}_{i\geq 1} \in (2^{\mathcal{X}})^\omega$ of subsets of $\mathcal{X}$, there is *at most* one integer $n_{\sigma,\mathbf{X}} \geq 1$ with $\mathsf{end} \in \sigma(X_1 \cdots X_{n_{\sigma,\mathbf{X}}})$. If none exists, we write $n_{\sigma,\mathbf{X}} = \infty$. To account for traces that do not contain end, we redefine $\mathsf{traces^f}(\sigma)$ as follows: $\{\pi^{\mathsf{f}}[\sigma, \mathbf{X}] \mid \mathbf{X} \in (2^{\mathcal{X}})^\omega$ and $n_{\sigma,\mathbf{X}} < \infty\}$. A strategy σ is an *α-strategy* if for every $\mathbf{X} \in (2^{\mathcal{X}})^\omega$, either $n_{\sigma,\mathbf{X}} < \infty$ or $\pi[\sigma, \mathbf{X}] \not\models \alpha$, i.e. σ terminates on every trace that satisfies α. A *winning strategy* (w.r.t. $\langle\mathcal{X}, \mathcal{Y}, \alpha, \varphi\rangle$) is an α-strategy such that $\pi \models \varphi$ for every $\pi \in \mathsf{traces^f}(\sigma)$. In other words, winning strategies are those that guarantee the satisfaction of the objective φ under the assumption that the environment behaves in a way that constraint α is satisfied. The realizability and synthesis problems for constrained $\mathsf{LTL_f}$ specifications are defined as before, using this notion of winning strategy.

Because the constraints are interpreted using infinite LTL semantics, we are now able to correctly handle liveness constraints ($\Diamond\psi$) and fairness constraints as studied in LTL synthesis ($\Box\Diamond\psi$) and FOND planning ($\Box\Diamond\psi_1 \rightarrow \Box\Diamond\psi_2$) (D'Ippolito, Rodríguez, and Sardiña 2018).

Example 1. Returning to our earlier example, consider the constrained synthesis problem with assumption Ψ^{cat}_{env} (which includes φ_{leaves}) and objective Ψ^{cat}_{robot}. The obvious strategy (vacuum dirty rooms as soon as they are cat-free) gives rise to a winning strategy, in which we output end if we manage to clean both rooms, and otherwise, produce an infinite trace without end in which Ψ^{cat}_{env} is not true. Trivial strategies that terminate immediately will *not* be winning strategies, as there will be infinite traces that satisfy the constraint but where the length-1 finite trace falsifies the objective.

We remark that if we are not careful about how we write the constraint α, we may unintentionally allow the agent to block the environment from fulfilling α.

Remark 1. Suppose that instead of using Equation 1 to encode the effects of actions, we employ the simpler $\Box((\kappa \wedge a) \rightarrow \bullet\beta)$. While intuitive, this alternative formulation does not properly encode FOND planning, as the specification may be realized in an unintended way: by performing multiple actions with conflicting effects, or a single action whose precondition is not satisfied, the agent can force the environment to satisfy a contradictory set of formulae β in the next state, causing the assumption to be violated.

Chatterjee, Henzinger, and Jobstmann (2008) discuss this phenomenon in the context of LTL synthesis, and suggest that a reasonable environment constraint is one which is *realizable for the environment*. We note that the constraints we considered in Section 3 all satisfy this property.

Correspondence with Finite LTL Synthesis We begin by observing that (plain) $\mathsf{LTL_f}$ synthesis is a special case of constrained $\mathsf{LTL_f}$ synthesis in which one uses the trivial constraint $\top$ for the environment assumption:

Theorem 1. *Every winning strategy σ for the $\mathsf{LTL_f}$ specification $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ is a winning strategy for the constrained $\mathsf{LTL_f}$ specification $\langle \mathcal{X}, \mathcal{Y}, \top, \varphi \rangle$, and vice-versa. In particular, $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ is realizable iff $\langle \mathcal{X}, \mathcal{Y}, \top, \varphi \rangle$ is realizable.*

A natural question is whether a reduction in the other direction exists. Indeed, it is well-known that in the infinite setting, assume-guarantee LTL synthesis[4] with an assumption α and objective φ corresponds to classical LTL synthesis w.r.t. $\alpha \rightarrow \varphi$ (that is, the two synthesis problems have precisely the same winning strategies). The following negative result shows that a simple reduction via implication does not work in the finite trace setting:

Theorem 2. *There exists an unrealizable constrained $\mathsf{LTL_f}$ specification $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$ such that the $\mathsf{LTL_f}$ specification $\mathcal{S}_\rightarrow = \langle \mathcal{X}, \mathcal{Y}, \alpha \rightarrow \varphi \rangle$ is realizable.*

Proof. Consider the constrained $\mathsf{LTL_f}$ specification $\mathcal{S} = \langle \{x, x'\}, \{y\}, \alpha, \varphi \rangle$ with $\alpha = \neg x \wedge \Diamond x$ and $\varphi = \Diamond(x' \wedge y)$. We claim that $\mathcal{S}$ is unrealizable. Indeed, take any $\mathbf{X} = X_1 X_2 \ldots$ such that $x \notin X_1$, $x \in X_2$, and $x' \notin X_i$ for all $i \geq 1$. Then no matter which strategy σ is used, the infinite trace $\pi[\sigma, \mathbf{X}]$ will satisfy α, and the induced finite trace $\pi^{\mathrm{f}}[\sigma, \mathbf{X}]$, if it exists, will falsify φ (as x' never holds).

Next consider $\mathcal{S}_\rightarrow = \langle \{x\}, \{y\}, \alpha \rightarrow \varphi \rangle$, and observe that $\alpha \rightarrow \varphi \equiv x \vee (\Box \neg x) \vee \Diamond(x' \wedge y)$. A simple winning strategy exists: output end in the first time step. Indeed, every induced trace has length 1 and hence trivially satisfies $x \vee \Box\neg x$. ◻

With the next theorem, we observe a more fundamental difficulty in reducing constrained $\mathsf{LTL_f}$ synthesis problems to standard $\mathsf{LTL_f}$ synthesis: winning strategies for constrained problems may need an *unbounded* number of time steps to realize the specification, a phenomenon that does not occur in standard $\mathsf{LTL_f}$ synthesis.

Theorem 3. *An $\mathsf{LTL_f}$ specification is realizable iff it admits a* bounded *winning strategy, i.e. a strategy for which there exists $B > 0$ such that every induced finite trace has length at most B. There exist realizable constrained $\mathsf{LTL_f}$ specifications that do not possess any bounded winning strategy.*

Proof. A straightforward examination of the $\mathsf{LTL_f}$ synthesis algorithm[5] in (De Giacomo and Vardi 2015) shows that when $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ is realizable, the produced strategy guarantees achievement of φ in a number of time steps bounded by the number of states in a DFA for φ.

For the second point, consider the constrained $\mathsf{LTL_f}$ specification $\mathcal{S} = \langle \{x\}, \{y\}, \Diamond x, \neg y \ \mathsf{U}\ (x \wedge y) \rangle$. Observe that $\mathcal{S}$ is realizable, as it suffices to output $\neg y$ until the first x is read, then output $\{y, \mathsf{end}\}$. Assume for a contradiction that there is a winning strategy σ for $\mathcal{S}$ and constant $B > 0$ such that $n_{\sigma,\mathbf{X}} \leq B$ for every $\mathbf{X} \in \mathcal{X}^\omega$. Define $\mathbf{X}^B = X_1^B X_2^B \ldots$ as follows: $X_i^B = \{x\}$ if $i = B + 1$ and $X_i^B = \emptyset$ otherwise. The induced trace $\pi = \pi^{\mathrm{f}}[\sigma, \mathbf{X}^B]$ has length at most B and hence does not contain x. It follows that $\pi \not\models \varphi$, contradicting our assumption that σ is a winning strategy. ◻

While the implication-based approach does not work in general, we show that it can be made to work for environment assumptions that belong to the safe fragment:

Theorem 4. *When α is a safe formula, the constrained $\mathsf{LTL_f}$ specification $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$ is realizable iff the $\mathsf{LTL_f}$ specification $\mathcal{S}' = \langle \mathcal{X}, \mathcal{Y}, \alpha' \rightarrow \varphi \rangle$ is realizable, where α' is obtained from α by replacing every occurrence of $\bigcirc\psi$ by $\bullet\psi$.*

Proof sketch. Let σ' be a winning strategy for $\langle \mathcal{X}, \mathcal{Y}, \alpha' \rightarrow \varphi \rangle$, with α a safe formula. To define a winning strategy σ for $\langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$, we set $\sigma(X_1 \cdots X_n)$ equal to
- $\sigma(X_1 \cdots X_n) \setminus \{\mathsf{end}\}$, when $\mathsf{end} \in \sigma(X_1 \cdots X_n)$ and $(X_1 \cup \sigma(X_1)) \ldots (X_n \cup \sigma(X_1 \cdots X_n)) \not\models \alpha'$;
- $\sigma(X_1 \cdots X_n)$, otherwise.

For the other direction, given a winning strategy σ for $\langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$, we can define a winning strategy σ' for $\langle \mathcal{X}, \mathcal{Y}, \alpha' \rightarrow \varphi \rangle$ by setting $\sigma'(X_1 \cdots X_n)$ equal to

- $\sigma(X_1 \cdots X_n) \cup \{\mathsf{end}\}$, if $(X_1 \cup \sigma(X_1)) \ldots (X_n \cup \sigma(X_1 \cdots X_n))$ is a bad prefix for α, and $\mathsf{end} \notin \sigma'(X_1 \cdots X_k)$ for $k < n$;
- $\sigma(X_1 \cdots X_n) \setminus \{\mathsf{end}\}$, if $\mathsf{end} \in \sigma'(X_1 \cdots X_k)$ for some $k < n$;
- $\sigma'(X_1 \cdots X_n) = \sigma(X_1 \cdots X_n)$, otherwise. ◻

[4]Here we refer to assume-guarantee synthesis as considered in (Chatterjee, Henzinger, and Jobstmann 2008; Almagor et al. 2017), where given a pair (α, φ), the aim is to construct a strategy such that every induced infinite trace either violates α or satisfies φ. This is different from the assume-guarantee synthesis of (Chatterjee and Henzinger 2007), in which N agents each have their own goals, and the objective is for each agent to satisfy its own goals.

[5]The algorithm can be easily modified to output end once φ has been satisfied to match our definition of strategy.

The following example shows that it is essential in the preceding theorem to use $\alpha' \to \varphi$ rather than $\alpha \to \varphi$:

Example 2. If we let $\alpha = \Box(\neg x \vee \bigcirc x)$ and $\varphi = \neg x \wedge y$, then the constrained specification $\langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$ is not realizable (as the environment can output x in the first step), but the $\mathsf{LTL_f}$ specification $\langle \mathcal{X}, \mathcal{Y}, \alpha \to \varphi \rangle$ is realizable with a strategy that outputs $\{y, \mathsf{end}\}$ in the first step. Indeed, if the environment outputs x, then $\neg\alpha \equiv \Diamond(x \wedge \neg\bigcirc x)$ holds in the induced length-1 trace; if we have $\neg x$ instead, then φ holds.

Note however that the negative result in the general case (Theorem 2) continues to hold if $\alpha' \to \varphi$ is used instead of $\alpha \to \varphi$, since the formulas in that proof do not involve $\bigcirc$.

Another interesting observation is the environment assumptions Ψ_{init} and Ψ_{eff} used to encode the initial state and action effects in planning are safe formulas. This explains why these constraints can be properly encoded in $\mathsf{LTL_f}$ using implication and $\bullet$ rather than $\bigcirc$. We note that if we encode planning using constrained $\mathsf{LTL_f}$ synthesis, then we can use $\bigcirc$ in the effect axioms, which is arguably more natural.

Reduction to LTL Synthesis Every $\mathsf{LTL_f}$ formula φ over $\mathcal{P}$ can be polynomially transformed into an LTL formula φ_{inf} over $\mathcal{P} \cup \{\mathsf{alive}\}$ such that $\pi \models \varphi_{\mathsf{inf}}$ iff $\pi' \models \varphi$ for some finite prefix $\pi' \sqsubseteq \pi$ (De Giacomo and Vardi 2013). Intuitively, alive holds for the duration of the (simulated) finite trace. Formally, $\varphi_{\mathsf{inf}} := \tau(\varphi) \wedge \mathsf{alive} \wedge (\mathsf{alive}\, \mathsf{U}\, (\Box\neg\mathsf{alive}))$, where:

$$\tau(p) = p \quad \tau(\neg\varphi) = \neg\tau(\varphi) \quad \tau(\varphi_1 \wedge \varphi_2) = \tau(\varphi_1) \wedge \tau(\varphi_2)$$
$$\tau(\bigcirc\varphi) = \bigcirc(\mathsf{alive} \wedge \tau(\varphi))\, \tau(\varphi_1\, \mathsf{U}\, \varphi_2) = \tau(\varphi_1)\, \mathsf{U}\, (\mathsf{alive} \wedge \tau(\varphi_2))$$

We extend this transformation as follows:

$$\psi_{\mathsf{end}} := \Box(\mathsf{end} \leftrightarrow \mathsf{alive} \wedge \bigcirc\neg\mathsf{alive}) \wedge \Box(\mathsf{end} \to \bigcirc\Box\neg\mathsf{end})$$
$$\psi_{\alpha,\varphi} := \psi_{\mathsf{end}} \wedge ((\alpha \vee \Diamond\mathsf{end}) \to \varphi_{\mathsf{inf}})$$

Here ψ_{end} forces the agent to trigger variable end when the end of the trace is simulated and also ensures that end occurs at most once. Formula $\psi_{\alpha,\varphi}$ ensures that φ_{inf} is satisfied – i.e., a finite trace that satisfies φ and ends is simulated – when either the environment assumption α holds or end occurs.

Theorem 5. *The constrained* $\mathsf{LTL_f}$ *specification* $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$ *is realizable iff* LTL *specification* $\mathcal{S}^\infty = \langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{alive}, \mathsf{end}\}, \psi_{\alpha,\varphi} \rangle$ *is realizable. Moreover, for every winning strategy* σ *for* $\mathcal{S}^\infty$, *the strategy* σ' *defined by* $\sigma'(X_1 \cdots X_n) := \sigma(X_1 \cdots X_n) \setminus \{\mathsf{alive}\}$ *is a winning strategy for* $\mathcal{S}$.

5 Algorithms for Constrained $\mathsf{LTL_f}$ Synthesis

LTL and $\mathsf{LTL_f}$ realizability are both 2EXP-complete (Pnueli and Rosner 1989; De Giacomo and Vardi 2015), and we can show the same holds for constrained $\mathsf{LTL_f}$ problems. The upper bound exploits the reduction to LTL (Theorem 5), and the lower bound is inherited from (plain) $\mathsf{LTL_f}$ synthesis, which is a special case of constrained $\mathsf{LTL_f}$ synthesis (Theorem 1).

Theorem 6. *Constrained* $\mathsf{LTL_f}$ *realizability (resp. synthesis) is 2EXP-complete (resp. in 2EXP).*

It follows from Theorem 6 that the reduction to infinite LTL realizability and synthesis yields worst-case optimal algorithms. However, we argue that the reduction to LTL does not provide a practical approach. Indeed, while LTL and $\mathsf{LTL_f}$ synthesis share the same worst-case complexity, recent experiments have shown that $\mathsf{LTL_f}$ is much easier to handle in practice (Zhu et al. 2017). Indeed, state-of-the-art approaches to LTL synthesis rely on first translating the LTL formula into a suitable infinite-word automata, then solving a two-player game on the resulting automaton. The computational bottleneck is the complex transformations of infinite-word automata, for which no efficient implementations exist. Recent approaches to $\mathsf{LTL_f}$ synthesis also adopt an automata-game approach, but $\mathsf{LTL_f}$ formulae require only finite-word automata (NFAs and DFAs), which can be manipulated more efficiently.

The preceding considerations motivate us to explore an alternative approach to constrained $\mathsf{LTL_f}$ synthesis, which involves a reduction to DBA games. Importantly, the DBA can be straightforwardly constructed from DFAs for the constraint and objective formulae, allowing us to sidestep the difficulties of manipulating infinite-word automata.

5.1 DBA for Constrained Specifications

For the rest of this section, we assume $\alpha = \alpha_s \wedge \alpha_c$, where α_s is a safe formula and α_c is a co-safe formula[6], both defined over $\mathcal{X} \cup \mathcal{Y}$. Safe and co-safe formulae are well-known LTL fragments (Kupferman and Vardi 2001) of proven utility. Safe formulas are prevalent in LTL specifications and a key part of the encoding of planning as $\mathsf{LTL_f}$ synthesis (see Section 3.3); the usefulness of co-safe formulas can be seen from our example (Section 3.4) and their adoption in work on robot planning (see e.g. (Lahijanian et al. 2015)).

Our aim is to construct a DBA that accepts infinite traces π over $2^{\mathcal{X}\cup\mathcal{Y}\cup\{\mathsf{end}\}}$ such that either (i) π contains a single occurrence of end which induces a finite prefix π' with $\pi' \models \varphi$, or (ii) π doesn't contain end and $\pi \not\models \alpha_s \wedge \alpha_c$. Such a DBA $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ can be defined by combining three DFAs: $\mathcal{A}_s = \langle 2^{\mathcal{P}}, Q_s, \delta_s, (q_0)_s, F_s \rangle$ accepts the bad prefixes of α_s; $\mathcal{A}_c = \langle 2^{\mathcal{P}}, Q_c, \delta_c, (q_0)_c, F_c \rangle$ accepts the good prefixes of α_c; and $\mathcal{A}_g = \langle 2^{\mathcal{P}}, Q_g, \delta_g, (q_0)_g, F_g \rangle$ accepts models of φ. Recall that these DFAs can be built in double-exponential time.

Formally, we let $\mathcal{A}_\varphi^{\alpha_s,\alpha_c} = \langle 2^{\mathcal{P}\cup\{\mathsf{end}\}}, Q, \delta, q_0, F \rangle$, where Q, q_0, and F are defined as follows:

- $Q := \big((Q_s \cup \{q_{\mathsf{bad}}\}) \times (Q_c \cup \{q_{\mathsf{good}}\}) \times Q_g\big) \cup \{q_\top, q_\bot\}$
- $q_0 := ((q_0)_s, (q_0)_c, (q_0)_g)$
- $F := \{(q_s, q_c, q_g) \in Q \mid q_s = q_{\mathsf{bad}} \text{ or } q_c \neq q_{\mathsf{good}}\} \cup \{q_\top\}$

For 'regular' symbols $\theta \in 2^{\mathcal{P}}$ (i.e., $\mathsf{end} \notin \theta$), we set $\delta((q_s, q_c, q_g), \theta) = (\delta_s^*(q_s, \theta), \delta_c^*(q_c, \theta), \delta_g(q_g, \theta))$ where:

$$\delta_s^*(q_s, \theta) = \begin{cases} q_{\mathsf{bad}}, & \text{if } q_s = q_{\mathsf{bad}} \text{ or } \delta_s(q_s, \theta) \in F_s \\ \delta_s(q_s, \theta), & \text{otherwise} \end{cases}$$

$$\delta_c^*(q_c, \theta) = \begin{cases} q_{\mathsf{good}}, & \text{if } q_c = q_{\mathsf{good}} \text{ or } \delta_c(q_c, \theta) \in F_c \\ \delta_c(q_c, \theta), & \text{otherwise} \end{cases}$$

For θ with $\mathsf{end} \in \theta$, we set $\delta((q_s, q_c, q_g), \theta) = q_\top$ if $\delta_g(q_g, \theta) \in F_g$, and $\delta((q_s, q_c, q_g), \theta) = q_\bot$ in all other cases. Accepting state $q_\top$ is quasi-absorbing: $\delta(q_\top, \theta) = q_\top$ when $\mathsf{end} \notin \theta$, and $\delta(q_\top, \theta) = q_\bot$ otherwise. This forces winning

[6]If we want to have *only* a safe (resp. co-safe) constraint, it suffices to use a trivial constraint $\alpha_c = \top$ (resp. $\alpha_s = \bot\, \mathsf{R}\, (p \vee \neg p)$).

strategies to output variable end at most once. Finally, $q_\perp$ is an absorbing state: $\delta(q_\perp, \theta) = q_\perp$ for every $\theta \in 2^{\mathcal{P}\cup\{\mathsf{end}\}}$.

Theorem 7. *The DBA $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ accepts infinite traces π such that either: (i) $\pi(1)\cdots\pi(n) \models \varphi$ and* end *occurs only in $\pi(n)$, or (ii) $\pi \not\models \alpha_s \wedge \alpha_c$ and* end *does not occur in π. $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ can be constructed in double-exponential time in $|\alpha_s| + |\alpha_c| + |\varphi|$.*

5.2 DBA Games

Once a specification has been converted into a DBA, realizability and synthesis can be reduced to DBA games. We briefly recall next the definition of such games and how winning strategies can be computed.

A *DBA (or Büchi) game* (see e.g. (Chatterjee, Henzinger, and Piterman 2006)) is a two-player game given by a tuple $\langle \mathcal{X}, \mathcal{Y}, \mathcal{A}\rangle$, where $\mathcal{X}$ and $\mathcal{Y}$ are disjoint finite sets of variables and $\mathcal{A}$ is a DBA with alphabet $2^{\mathcal{X}\cup\mathcal{Y}}$. A *play* is an infinite sequence of rounds, where in each round, Player I selects $X_i \subseteq \mathcal{X}$, then Player II selects $Y_i \subseteq \mathcal{Y}$. A play is winning if it yields a word $(X_1 \cup Y_1)(X_2 \cup Y_2)\ldots$ that belongs to $\mathcal{L}(\mathcal{A})$. A game is *winning* if there exists a strategy $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ such that for every infinite sequence $X_1X_2\ldots \in \mathcal{X}^\omega$, the word $(X_1 \cup \sigma(X_1))(X_2 \cup \sigma(X_1X_2))\ldots$ obtained by following σ belongs to $\mathcal{L}(\mathcal{A})$. In this case, we call σ a *winning strategy*.

Existence of a winning strategy for a DBA game $\mathcal{G} = \langle \mathcal{X}, \mathcal{Y}, \mathcal{A}\rangle$ based upon $\mathcal{A} = \langle 2^{\mathcal{X}\cup\mathcal{Y}}, Q, \delta, q_0, F\rangle$ can be determined by computing the winning region of $\mathcal{G}$. This is done in two steps. First, we compute the set $RA(\mathcal{G})$ of *recurring accepting* states, i.e. those $q \in F$ such that Player II has a strategy from state q to revisit F infinitely often. Next, we define the *winning region* $Win(\mathcal{G})$ of $\mathcal{G}$ as those states in $q \in Q$ for which Player II has a strategy for reaching a state in $RA(\mathcal{G})$. The sets $RA(\mathcal{G})$ and $Win(\mathcal{G})$ can be computed in polynomial time by utilizing the *controllable predecessor* operator: $\mathsf{CPre}(S) = \{q \in Q \mid \forall X \subseteq \mathcal{X}\, \exists Y \subseteq \mathcal{Y} : \delta(q, X \cup Y) \in S\}$. We set $\mathsf{Reach}^0(S) = S$ and $\mathsf{Reach}^{i+1}(S) = \mathsf{Reach}^i(S) \cup \mathsf{CPre}(\mathsf{Reach}^i(S))$. Intuitively, $\mathsf{Reach}^i(S)$ contains those states from which Player II has a strategy for reaching (or returning to) S in at most i rounds. The limit $\lim_i \mathsf{Reach}^i(S)$ exists because $\mathsf{Reach}^i(S) \subseteq \mathsf{Reach}^{i+1}(S)$, and convergence is achieved in a finite number of iterations bounded by $|Q|$. To compute $RA(\mathcal{G})$, we set $S_1 = F$ and let $S_{k+1} = S_k \cap \lim_i \mathsf{Reach}^i(S_k)$. The set S_k contains those accepting states from which Player II has a strategy for visiting S_k no less than k times. The limit $\lim_k S_k$ exists because $S_k \subseteq S_{k+1}$, and convergence is achieved in a finite number of iterations bounded by $|F|$. $RA(\mathcal{G})$ is the finite limit of S_k, and the set $Win(\mathcal{G})$ is then the finite limit of $\mathsf{Reach}^i(RA(\mathcal{G}))$. It is easy to see that $Win(\mathcal{G})$ can be computed in polynomial time w.r.t. the size of the DBA $\mathcal{A}$. The following well-known result shows how we can use $Win(\mathcal{G})$ to decide if $\mathcal{G}$ is winning.

Theorem 8. *$\mathcal{G}$ is winning iff $q_0 \in Win(\mathcal{G})$.*

We sketch the proof of the right-to-left implication here, since it will be needed for later results. We suppose that $q_0 \in Win(\mathcal{G})$ and show how to construct a transducer $\mathcal{T}_\mathcal{G}$ that implements a winning strategy. Intuitively, the transducer's output function ensures that the transducer stays within $Win(\mathcal{G})$, always reducing the 'distance' to $RA(\mathcal{G})$. More precisely, we can define $\mathcal{T}_\mathcal{G}$ as $\langle 2^{\mathcal{X}}, 2^{\mathcal{Y}}, Q, \delta', \omega, q_0\rangle$, where: the set of states Q and initial state q_0 are the same as for the DBA $\mathcal{A}$, and the transition function δ' mirrors the transition function δ of $\mathcal{A}$: $\delta'(q, X) = \delta(q, X \cup \omega(q, X))$. We define the output function ω as follows:

- Case 1: there exists Y^* such that $\delta(q, X \cup Y^*) \in Win(\mathcal{G})$. In this case, we let $\omega(q, X)$ be any[7] $Y \in 2^{\mathcal{Y}}$ such that (a) $\delta(q, X \cup Y) \in \mathsf{Reach}^{i+1}(RA(\mathcal{G}))$, and (b) there is no Y' with $\delta(q, X \cup Y') \in \mathsf{Reach}^i(RA(\mathcal{G}))$.
- Case 2: no such Y^* exists. We let $\omega(q, X)$ be any $Y \in 2^{\mathcal{Y}}$.

According to this definition, after reading X, the transducer $\mathcal{T}_\mathcal{G}$ chooses an output symbol Y that allows the underlying automaton $\mathcal{A}$ to transition from the current state via $X \cup Y$ to a state in $Win(\mathcal{G})$ (if some such symbol exists). Moreover, among the immediately reachable winning states, preference is given to those that are closest to $RA(\mathcal{G})$, i.e. those belonging to $\mathsf{Reach}^i(RA(\mathcal{G}))$ for the minimal value i.

5.3 Constrained $\mathsf{LTL_f}$ Synthesis via DBA Games

Given a constrained $\mathsf{LTL_f}$ synthesis specification $\langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi\rangle$ with $\alpha = \alpha_s \wedge \alpha_c$, we proceed as follows:

1. Construct the DBA game $\mathcal{G}_\varphi^{\alpha_s,\alpha_c} = \langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{end}\}, \mathcal{A}_\varphi^{\alpha_s,\alpha_c}\rangle$.
2. Determine whether $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ is winning: build $Win(\mathcal{G}_\varphi^{\alpha_s,\alpha_c})$ and check whether $((q_0)_s, (q_0)_c, (q_0)_g) \in Win(\mathcal{G}_\varphi^{\alpha_s,\alpha_c})$.
3. If $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ is not winning, return 'unrealizable'.
4. Otherwise, compute a winning strategy for $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ using the transducer from Section 5.2.

Using Theorems 7 and 8, we can show that this method is correct and yields optimal complexity:

Theorem 9. *Consider a constrained $\mathsf{LTL_f}$ specification $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha_s \wedge \alpha_c, \varphi\rangle$ where α_s (resp. α_c) is a safe (resp. co-safe) formula. Then:*

- *$\mathcal{S}$ is realizable iff the DBA game $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ is winning;*
- *Every winning strategy for $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ is a winning strategy for $\mathcal{S}$, and vice-versa;*
- *Deciding whether $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ is winning, and constructing a winning strategy when one exists, can be done in 2EXP.*

6 Synthesis of High-Quality Strategies

This section explores the use of a quantitative specification language to compare strategies based upon how well they satisfy the specification. We adopt the $\mathsf{LTL_f}[\mathcal{F}]$ language from (Almagor, Boker, and Kupferman 2016) and propose a new more refined way of defining optimal strategies.

6.1 The Temporal Logic $\mathsf{LTL_f}[\mathcal{F}]$

We recall here the language $\mathsf{LTL_f}[\mathcal{F}]$ proposed by Almagor, Boker, and Kupferman (2016). The basic idea is that instead of a formula being either totally satisfied or totally violated by a trace, a value between 0 and 1 will indicate its *degree of satisfaction*. In order to allow for different ways of aggregating formulae, the basic LTL syntax is augmented with a set $\mathcal{F} \subseteq \{f : [0,1]^k \to [0,1] \mid k \in \mathbb{N}\}$ of functions, with the

[7] Several Y may satisfy the conditions, and choosing *any* such Y yields a suitable transducer. Alternatively, one can use nondeterministic transducers (called *strategy generators* in (De Giacomo and Vardi 2015)) to encode a family of deterministic transducers.

choice of which functions to include in $\mathcal{F}$ being determined by the application at hand.

Formally, the set of $\mathsf{LTL_f}[\mathcal{F}]$ formulae is obtained by adding $f(\varphi_1, \ldots, \varphi_k)$ to the grammar for φ, for every $f \in \mathcal{F}$. We assign a *satisfaction value* to every $\mathsf{LTL_f}[\mathcal{F}]$ formula, finite trace π, and time step $1 \leq i \leq |\pi|$, as follows[8]:

$$[\![\pi, \top]\!]_i = 1 \quad [\![\pi, \bot]\!]_i = 0 \quad [\![\pi, p]\!]_i = \begin{cases} 1 & \text{if } p \in \pi(i) \\ 0 & \text{otherwise} \end{cases}$$

$$[\![\pi, \neg\varphi]\!]_i = 1 - [\![\pi, \varphi]\!]_i$$

$$[\![\pi, \varphi_1 \wedge \varphi_2]\!]_i = \min\{[\![\pi, \varphi_1]\!]_i, [\![\pi, \varphi_2]\!]_i\}$$

$$[\![\pi, \bigcirc\varphi]\!]_i = [\![\pi, \varphi]\!]_{i+1}$$

$$[\![\pi, f(\varphi_1, \ldots, \varphi_k)]\!]_i = f([\![\pi, \varphi_1]\!]_i, \ldots, [\![\pi, \varphi_k]\!]_i)$$

$$[\![\pi, \varphi_1 \mathbin{\mathsf{U}} \varphi_2]\!]_i = \max_{i \leq i' \leq |\pi|} \left\{ \min \left\{ [\![\pi, \varphi_2]\!]_{i'}, \min_{i \leq j < i'} \{[\![\pi, \varphi_1]\!]_j\} \right\} \right\}$$

The (satisfaction) value of φ on π, written $[\![\pi, \varphi]\!]$, is $[\![\pi, \varphi]\!]_1$.

We define $V(\varphi) \subseteq [0, 1]$ as the set of values $[\![\pi, \varphi]\!]_i$, ranging over all traces π and steps $1 \leq i \leq |\pi|$. The following proposition, proven by (Almagor, Boker, and Kupferman 2016), shows that an $\mathsf{LTL_f}[\mathcal{F}]$ formula can take on only exponentially many different values.

Proposition 1. *For every $\mathsf{LTL_f}[\mathcal{F}]$ formula φ, $|V(\varphi)| \leq 2^{|\varphi|}$.*

The functions f allow us to capture a diversity of methods for combining a set of potentially competing objectives (including classical preference aggregation methods like weighted sums and lexicographic ordering).

Example 3. For illustration purposes, consider two variants of our robot vacuum example, with specifications φ_1 and φ_2, in which the goal $\Diamond(clean(LR) \wedge clean(BR))$ is replaced by $\Diamond(clean(LR))$ and $\Diamond(clean(BR))$, respectively. We can include in $\mathcal{F}$ a binary weighted sum operator $sum_{0.3,0.7}$, where the satisfaction value of $sum_{0.3,0.7}(\varphi_1, \varphi_2)$ on trace π is 0.3 if $\pi \models \varphi_1 \wedge \neg\varphi_2$, 0.7 if $\pi \models \neg\varphi_1 \wedge \varphi_2$, 1 if $\pi \models \varphi_1 \wedge \varphi_2$, and zero otherwise. We can thus express that we'd like to clean both rooms, but give priority to the bedroom.

6.2 Defining Optimal Strategies

Henceforth, we consider a *constrained synthesis* $\mathsf{LTL_f}[\mathcal{F}]$ *problem* $\langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$, defined as before except that now φ is an $\mathsf{LTL_f}[\mathcal{F}]$ formula. Such formulae assign satisfaction values to traces, allowing us to rank traces according to the extent to which they satisfy the expressed preferences. It remains to lift this preference order to strategies.

Perhaps the most obvious way to rank strategies is to consider the minimum value of any trace induced by the strategy, preferring strategies that can guarantee the highest worst-case value. This is the approach adopted by (Almagor, Boker, and Kupferman 2016) for $\mathsf{LTL}[\mathcal{F}]$ synthesis. We formalize it for constrained $\mathsf{LTL_f}[\mathcal{F}]$ synthesis as follows:

Definition 1. The *best guaranteed value* of strategy σ, denoted $\mathsf{bgv}(\sigma)$, is the minimum value of $[\![\pi, \varphi]\!]$ over all $\pi \in \mathsf{traces^f}(\sigma)$ (or 0 if $\mathsf{traces^f}(\sigma) = \emptyset$). A strategy σ is *bgv-optimal w.r.t.* (α, φ) if it is a α-strategy and no α-strategy σ' exists with $\mathsf{bgv}(\sigma') > \mathsf{bgv}(\sigma)$.

[8]We omit $\vee$ and R, as they can be defined using $\neg$, $\wedge$, and U.

Optimizing for the best guaranteed value seems natural, but can be insufficiently discriminative. Consider a simple scenario with $\mathcal{X} = \{x\}$ and $\mathcal{Y} = \{y\}$. If the environment plays x, then we get value 0 no matter what, and if $\neg x$ is played, a value of 1 is achieved by playing y, and 0 if $\neg y$ is played. Clearly, we should prefer to play y after $\neg x$, yet the strategy that plays $\neg y$ following $\neg x$ is bgv-optimal, since like every strategy, its bgv is 0. This motivates us to introduce a stronger, context-aware, notion of optimality:

Definition 2. Given a strategy σ, trace $\pi \in \mathsf{ptraces}(\sigma)$ that does not contain end, and $X \in 2^{\mathcal{X}}$, the *best guaranteed value of σ starting from $\pi{\cdot}X$*, written $\mathsf{bgv}_{\pi,X}(\sigma)$, is the minimum of $[\![\pi', \varphi]\!]$ over all traces $\pi' \in \mathsf{traces^f}(\sigma)$ such that $\pi \cdot (X \cup Y) \sqsubseteq \pi'$ for some $Y \in \mathcal{Y}$ (or 0 if no such trace exists). A strategy σ is a *strongly bgv-optimal w.r.t.* (α, φ) if it is an α-strategy, and there is no α-strategy σ', trace $\pi \in \mathsf{ptraces}(\sigma) \cap \mathsf{ptraces}(\sigma')$ without end, and $X \in 2^{\mathcal{X}}$ such that $\mathsf{bgv}_{\pi,X}(\sigma') > \mathsf{bgv}_{\pi,X}(\sigma)$.

Strongly bgv-optimal strategies take advantage of any favorable situation during execution to improve the best worst-case value. In the preceding example, they allow us to say that the first strategy is better than the second.

7 Algorithms: High-Quality $\mathsf{LTL_f}$ Synthesis

In this section, we present novel techniques to compute bgv-optimal and strongly bgv-optimal strategies for a constrained $\mathsf{LTL_f}[\mathcal{F}]$ synthesis problem $\langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$. As in Section 5.1, we focus on the case where α is a conjunction $\alpha_s \wedge \alpha_c$ of safe and co-safe formulae.

7.1 Automaton for $\mathsf{LTL_f}[\mathcal{F}]$

It has been shown in (Almagor, Boker, and Kupferman 2016) how to construct, for a given $\mathsf{LTL_f}[\mathcal{F}]$ formula φ and set of values $\mathcal{V} \subseteq [0, 1]$, an NFA $\mathcal{A}_{\varphi,\mathcal{V}} = \langle 2^{\mathcal{P}}, Q, \delta, Q_0, F \rangle$ that accepts finite traces π with $[\![\pi, \varphi]\!] \in \mathcal{V}$. We briefly recall the construction here. We denote by $sub(\varphi)$ the set of subformulas of φ, and let C_φ be the set of functions $g : sub(\varphi) \to [0, 1]$ such that $g(\psi) \in V(\psi)$ for all $\psi \in sub(\varphi)$. Q contains all *consistent* functions in C_φ, where a function g is consistent if, for every $\psi \in sub(\varphi)$, the following hold:

- if $\psi = \top$, then $g(\psi) = 1$, and if $\psi = \bot$, then $g(\psi) = 0$
- if $\psi \in \mathcal{P}$ then $g(\psi) \in \{0, 1\}$
- if $\psi = f(\psi_1, \ldots, \psi_k))$, then $g(\psi) = f(g(\psi_1), \ldots, g(\psi_k))$

The transition function δ is such that $g' \in \delta(g, \sigma)$ whenever:

- $\sigma = \{p \in \mathcal{P} \mid g(p) = 1\}$
- $g(\bigcirc\psi_1) = g'(\psi_1)$ for every $\bigcirc\psi_1 \in sub(\varphi))$
- $g(\psi_1 \mathbin{\mathsf{U}} \psi_2) = \max\{g(\psi_2), \min\{g(\psi_1), g'(\psi_1 \mathbin{\mathsf{U}} \psi_2)\}\}$ for every $\psi_1 \mathbin{\mathsf{U}} \psi_2 \in sub(\varphi)$

Finally, the set of initial states is $Q_0 = \{q \in Q \mid g(\varphi) \in \mathcal{V}\}$, and $F = \{g \mid g(\psi_2) = g(\psi_1 \mathbin{\mathsf{U}} \psi_2)$ for all $\psi_1 \mathbin{\mathsf{U}} \psi_2 \in sub(\varphi)\} \cap \{g \mid g(\bigcirc\psi) = 0$ for all $\bigcirc\psi \in sub(\varphi)\}$.

The NFA $\mathcal{A}_{\varphi,\mathcal{V}}$ can be constructed in single-exponential time, and $\mathcal{L}(\mathcal{A}_{\varphi,\mathcal{V}}) = \{\pi \mid [\![\pi, \varphi]\!] \in \mathcal{V}\}$ (Almagor, Boker, and Kupferman 2016). By determinizing $\mathcal{A}_{\varphi,\mathcal{V}}$, we obtain a DFA $\hat{\mathcal{A}}_{\varphi,\mathcal{V}}$ that accepts the same language and can be constructed in double-exponential time. In what follows, $\mathcal{V}$ will always take the form $[b, 1]$, so we'll use $\hat{\mathcal{A}}_{\varphi \geq b}$ in place of $\hat{\mathcal{A}}_{\varphi,[b,1]}$.

7.2 Synthesis of bgv-optimal strategies

We describe how to construct a bgv-optimal strategy. First note that given $b \in [0, 1]$, we can construct a DBA $\mathcal{A}^{\alpha}_{\varphi \geq b}$ that recognizes traces such that either (i) $\alpha = \alpha_s \wedge \alpha_c$ is violated and end does not occur, or (ii) end occurs exactly once and the induced finite trace π is such that $[\![\pi, \varphi]\!] \geq b$. Indeed, we simply reuse the construction from Section 5.1, replacing the DFA $\mathcal{A}_g$ with the DFA $\hat{\mathcal{A}}_{\varphi \geq b}$. We next observe that an α-strategy σ with $\mathsf{bgv}(\sigma) \geq b$ exists iff the DBA game $\langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{end}\}, \mathcal{A}^{\alpha}_{\varphi \geq b} \rangle$ is winning. Thus, by iterating over the values in $V(\varphi)$ in descending order, we can determine the maximal b^* for which an α-strategy σ with $\mathsf{bgv}(\sigma) \geq b^*$ exists. A bgv-optimal strategy can be computed by constructing a winning strategy for the DBA game $\langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{end}\}, \mathcal{A}^{\alpha}_{\varphi \geq b^*} \rangle$, using the approach in Section 5.2. As there are only exponentially many values in $V(\varphi)$ (Prop. 1), the overall construction takes double-exponential time.

Theorem 10. *A bgv-optimal strategy can be constructed in double-exponential time.*

7.3 Synthesis of strongly bgv-optimal strategies

To compute a strongly bgv-optimal stategy, we build a transducer that runs in parallel DBAs $\mathcal{A}^{\alpha}_{\geq b}$ for different values b, and selects outputs symbols so as to advance within the 'best' applicable winning region. This idea can be formalized as follows. As in Section 7.2, we first determine the maximal $b^* \in V(\varphi)$ for which a α-strategy σ with $\mathsf{bgv}(\sigma) \geq b^*$ exists, and set $B = V(\varphi) \cap [b^*, 1]$. In the process, we will compute, for each $b \in B$, the sets $Win(\mathcal{G}_b)$ and $RA(\mathcal{G}_b)$ for the DBA game $\mathcal{G}_b = \langle \mathcal{X}, \mathcal{Y}, \mathcal{A}^{\alpha}_{\geq b} \rangle$ based on the DBA $\mathcal{A}^{\alpha}_{\geq b} = \langle 2^{\mathcal{P} \cup \{\mathsf{end}\}}, Q_b, \delta_b, q^b_0, F_b \rangle$. In the sequel, we will assume that the elements of B are ordered as follows: $b_1 < b_2 \ldots < b_m$, with $b^* = b_1$ and $b_m = 1$.

We now proceed to the definition of the desired transducer $\mathcal{T}^{\mathsf{str}} = \langle 2^{\mathcal{X}}, 2^{\mathcal{Y} \cup \{\mathsf{end}\}}, Q_{\mathsf{str}}, \delta_{\mathsf{str}}, \omega_{\mathsf{str}}, q^{\mathsf{str}}_0 \rangle$, obtained by taking the cross product of the set of DBAs $\mathcal{A}^{\alpha}_{\geq b}$ with $b \in B$, in order to keep track of the current states in these automata:

- $q^{\mathsf{str}}_0 = (q^{b_1}_0, q^{b_2}_0, \ldots, q^{b_m}_0)$ and $Q_{\mathsf{str}} = Q_{b_1} \times \ldots \times Q_{b_m}$
- $\delta_{\mathsf{str}}((q_1, \ldots, q_m), X) = (\delta_{b_1}(q_1, X \cup Y), \ldots, \delta_{b_m}(q_m, X \cup Y))$, where $Y = \omega_{\mathsf{str}}((q_1, q_2, \ldots, q_m), X)$

After reading X, the output function identifies the maximal value $b \in B$ such that current state q_b of $\mathcal{A}^{\alpha}_{\geq b}$ can transition, via some symbol $X \cup Y$, into a state in $Win(\mathcal{G}_b)$, and it returns the same output as the transducer $\mathcal{T}_{\mathcal{G}_b}$ in state q_b:

- $\omega_{\mathsf{str}}((q_1, \ldots, q_m), X) = \omega_b(q_b, X)$, where $b = \max(\{v \in B \mid \exists Y\, \delta_v(q_v, X \cup Y) \in Win(\mathcal{G}_v)\})$

We note that the transducer $\mathcal{T}_{\mathcal{G}_b}$ can be defined as in Section 5.2 even when $q^b_0 \notin Win(\mathcal{G}_b)$, but it only returns 'sensible' outputs when it transitions to $Win(\mathcal{G}_b)$.

Theorem 11. *$\mathcal{T}^{\mathsf{str}}$ implements a strongly bgv-optimal strategy and can be constructed in double-exponential time.*

8 Discussion and Concluding Remarks

It has been widely remarked in the (infinite) LTL synthesis literature that environment assumptions are ubiquitous: the existence of winning strategies is almost always predicated on some kind of environment assumption. This was observed in the work of (Chatterjee, Henzinger, and Jobstmann 2008), motivating the introduction of the influential assume-guarantee synthesis model, and in work on rational synthesis (Fisman, Kupferman, and Lustig 2010), where the environment is assumed to act as a rational agent, and synthesis necessitates finding a Nash equilibrium. Interesting reflections on the role of assumptions in LTL synthesis, together with a survey of the relevant literature, can be found in (Bloem et al. 2014).

In this paper, we explored the issue of handling environment assumptions in $\mathsf{LTL_f}$ synthesis (De Giacomo and Vardi 2013), the counterpart of LTL synthesis for programs that terminate. Our starting point was the observation that the standard approach to handling assumptions in LTL synthesis (via logical implication) fails in the finite-trace setting. This led us to propose an extension of $\mathsf{LTL_f}$ synthesis that explicitly accounts for environment assumptions. The key insight underlying the new model of *constrained* $\mathsf{LTL_f}$ *synthesis* is that while the synthesized program must realize the objective in a finite number of steps, the environment continues to exist after the program terminates, so environment assumptions should be interpreted under infinite LTL semantics.

We studied the relationships holding between constrained $\mathsf{LTL_f}$ synthesis and (standard) $\mathsf{LTL_f}$ and LTL synthesis. In particular, we identified a fundamental difficulty in reducing constrained $\mathsf{LTL_f}$ synthesis to $\mathsf{LTL_f}$ synthesis – the former problem can require unbounded strategies, while bounded strategies suffice for the latter. Nevertheless, when the constraints were restricted to the safe LTL fragment, a reduction from constrained $\mathsf{LTL_f}$ synthesis to $\mathsf{LTL_f}$ synthesis is possible. Interestingly, this explains why planning – more naturally conceived as a constrained $\mathsf{LTL_f}$ synthesis problem – can also be encoded as $\mathsf{LTL_f}$ synthesis. The connection between synthesis and planning has been remarked in several works (see e.g., (De Giacomo and Vardi 2015; D'Ippolito, Rodríguez, and Sardiña 2018; Camacho et al. 2018a; 2018b; 2018c)). We also showed how to reduce constrained $\mathsf{LTL_f}$ synthesis to (infinite) LTL synthesis, which provides a worst-case optimal means of solving constrained $\mathsf{LTL_f}$ synthesis problems, in the general case, using (infinite) LTL synthesis tools. In the case where our constraint is comprised of a conjunction of safe and co-safe formulae, we showed that the constrained $\mathsf{LTL_f}$ synthesis problem can be reduced to DBA games and the winning strategy determined from the winning region. What makes our approach interesting is that the DBA is constructed via manipulation of DFAs, much easier to handle in practice than infinite-word automata.

We next turned our attention to the problem of augmenting constrained $\mathsf{LTL_f}$ synthesis with quality measures. We were motivated by practical concerns surrounding the ability to differentiate and synthesize high-quality strategies in settings where we may have a collection of mutually unrealizable objective formulae and alternative strategies of differing quality. Our work builds on results for the infinite case, e.g., (Almagor, Boker, and Kupferman 2016; Almagor et al. 2017; Kupferman 2016) with and without environment assumptions. We adopted $\mathsf{LTL_f}[\mathcal{F}]$ as our language for specifying quality measures. While the syntax of $\mathsf{LTL_f}[\mathcal{F}]$ is utilitarian, many more compelling preference languages are reducible to this core language. We defined two different notions of optimal strategies – bgv-optimal and strongly bgv-optimal. The former adapts a similar definition in (Almagor, Boker, and Kupferman 2016) and the latter originates with us. We focused again on assumptions that can be expressed as conjunctions of safe and co-safe formulae and provided algorithms to compute bgv- and strongly bgv-optimal strategies with optimal (2EXP) complexity.

Proper handling of environment assumptions and quality measures, together with the design of efficient algorithms for such richer specifications, is essential to putting $\mathsf{LTL_f}$ synthesis into practice. The present paper makes several important advances in this direction and also suggests a number of interesting topics for future work including: the study of other types of assumptions in the finite-trace setting (e.g. rational synthesis), the exploitation of more compelling KR languages for specifying preferences, and the exploration of further ways of comparing and ranking strategies (perhaps incorporating notions of cost or trace length).

Relation to Conference Version

This paper appears in the Proceedings of the 16th International Conference on Knowledge Representation and Reasoning (KR 2018) without the appendix proofs. The body of this paper is the same as the KR 2018 paper, except that a minor typographic error in the translation of LTL into $\mathsf{LTL_f}$ (the definition of $\tau(\varphi_1 \mathsf{U} \varphi_2)$ on page 6), which originally appeared in (De Giacomo and Vardi 2013) and was repeated in the KR 2018 paper, has been corrected.

Acknowledgements

This work was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the ANR project GoAsQ (ANR-15-CE23-0022).

References

Almagor, S.; Kupferman, O.; Ringert, J. O.; and Velner, Y. 2017. Quantitative assume guarantee synthesis. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV)*, 353–374.

Almagor, S.; Boker, U.; and Kupferman, O. 2016. Formally reasoning about quality. *Journal of the ACM (JACM)* 63(3):24:1–24:56.

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence* 116(1-2):123–191.

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2009. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence* 173(5-6):593–618.

Bloem, R.; Ehlers, R.; Jacobs, S.; and Könighofer, R. 2014. How to handle assumptions in synthesis. In *Proceedings of the 3rd Workshop on Synthesis (SYNT)*, 34–50.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018a. Finite LTL synthesis as planning. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS)*, 29–38.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018b. Synthesizing controllers: On the correspondence between LTL synthesis and non-deterministic planning. In *Advances in Artificial Intelligence - Proceedings of the 31st Canadian Conference on Artificial Intelligence*, 45–59.

Camacho, A.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2018c. LTL realizability via safety and reachability games. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 4683–4691.

Chatterjee, K., and Henzinger, T. A. 2007. Assume-guarantee synthesis. In *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 261–275.

Chatterjee, K.; Henzinger, T. A.; and Jobstmann, B. 2008. Environment assumptions for synthesis. In *Proceedings of the 19th International Conference on Concurrency Theory (CONCUR)*, 147–161.

Chatterjee, K.; Henzinger, T.; and Piterman, N. 2006. Algorithms for Büchi games. In *GDV Workshop*. Available on Arxiv (arXiv:0805.2620).

Church, A. 1957. Applications of recursive arithmetic to the problem of circuit synthesis. *Summaries of the Summer Institute of Symbolic Logic, Cornell University 1957* 1:3–50.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 854–860.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 1558–1564.

D'Ippolito, N.; Rodríguez, N.; and Sardiña, S. 2018. Fully observable non-deterministic planning as assumption-based reactive synthesis. *Journal of Artificial Intelligence Research* 61:593–621.

Fisman, D.; Kupferman, O.; and Lustig, Y. 2010. Rational synthesis. In *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 190–204.

Kupferman, O., and Vardi, M. Y. 2001. Model checking of safety properties. *Formal Methods in System Design* 19(3):291–314.

Kupferman, O. 2016. On high-quality synthesis. In *Proceedings of the 11th International Computer Science Symposium in Russia*, 1–15.

Lahijanian, M.; Almagor, S.; Fried, D.; Kavraki, L. E.; and Vardi, M. Y. 2015. This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 3664–3671.

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages (POPL)*, 179–190.

Pnueli, A. 1977. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, 46–57.

Rabin, M. O., and Scott, D. S. 1959. Finite automata and their decision problems. *IBM Journal of Research and Development* 3(2):114–125.

Sistla, A. P. 1994. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing* 6(5):495–512.

Zhu, S.; Tabajara, L. M.; Li, J.; Pu, G.; and Vardi, M. Y. 2017. Symbolic LTLf synthesis. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 1362–1369.

Proofs

Theorem 4. When α is a safe formula, the constrained $\mathsf{LTL_f}$ specification $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$ is realizable iff the $\mathsf{LTL_f}$ specification $\mathcal{S}' = \langle \mathcal{X}, \mathcal{Y}, \alpha' \rightarrow \varphi \rangle$ is realizable, where α' is obtained from α by replacing every occurrence of $\bigcirc\psi$ by $\bullet\psi$.

Proof. Let σ' be a winning strategy for $\mathcal{S}' = \langle \mathcal{X}, \mathcal{Y}, \alpha' \rightarrow \varphi \rangle$, with α a safe formula. We define a strategy σ for $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$ by setting $\sigma(X_1 \cdots X_n)$ equal to

- $\sigma'(X_1 \cdots X_n) \setminus \{\mathsf{end}\}$, when $\mathsf{end} \in \sigma'(X_1 \cdots X_n)$ and $(X_1 \cup \sigma'(X_1)) \ldots (X_n \cup \sigma'(X_1 \cdots X_n)) \not\models \alpha'$;
- $\sigma'(X_1 \cdots X_n)$, otherwise.

To show that σ is a winning strategy, take some $\mathbf{X} \in (2^{\mathcal{X}})^\omega$. Then $n_{\sigma',\mathbf{X}} < \infty$, and the finite trace $\pi' = \pi^{\mathsf{f}}[\sigma', \mathbf{X}]$ is such that (i) $\pi' \models \neg\alpha'$, or (ii) $\pi' \models \varphi$ If (i) holds, then $n_{\sigma,\mathbf{X}} = \infty$ (since we will remove end), and $\pi = \pi[\sigma, \mathbf{X}]$ contains π' as a prefix. We can then use the $\mathsf{LTL_f}$ equivalence $\neg\bullet\varphi \equiv \bigcirc\neg\varphi$ and the fact that α is safe to derive $\pi \models \neg\alpha$. If (ii) holds (and (i) does not), then it follows from the definition of σ that $n_{\sigma,\mathbf{X}} = n_{\sigma',\mathbf{X}}$ and $\pi = \pi^{\mathsf{f}}[\sigma, \mathbf{X}] = \pi^{\mathsf{f}}[\sigma', \mathbf{X}]$, so $\pi \models \varphi$.

For the other direction, let σ be a winning strategy for $\mathcal{S}$. Define a strategy σ' for $\mathcal{S}'$ by setting $\sigma'(X_1 \cdots X_n)$ equal to

- $\sigma(X_1 \cdots X_n) \cup \{\mathsf{end}\}$, if $(X_1 \cup \sigma(X_1)) \ldots (X_n \cup \sigma(X_1 \cdots X_n))$ is a bad prefix for α, and $\mathsf{end} \notin \sigma'(X_1 \cdots X_k)$ for $k < n$;
- $\sigma(X_1 \cdots X_n) \setminus \{\mathsf{end}\}$, if $\mathsf{end} \in \sigma'(X_1 \cdots X_k)$ for some $k < n$;
- $\sigma'(X_1 \cdots X_n) = \sigma(X_1 \cdots X_n)$, otherwise

Basically, σ' mimics σ and outputs end as soon as a bad prefix is reached or σ terminates the execution due to satisfaction of φ, whichever situation occurs first. The construction of σ' is such that executions always terminate – which is not necessarily true for σ – and induce finite traces that satisfy the $\mathsf{LTL_f}$ formula $\alpha' \rightarrow \varphi$.

To show that σ' is a winning strategy for $\mathcal{S}'$, take some $\mathbf{X} \in (2^{\mathcal{X}})^\omega$. Consider the infinite induced trace $\pi = \pi[\sigma, \mathbf{X}]$, and let $k_{\mathbf{X}}$ be the minimum k such that $\pi(1) \ldots \pi(k)$ is a bad prefix for α ($k_{\mathbf{X}} = \infty$ if π has no bad prefix). We examine these three cases separately: (i) $n_{\sigma,\mathbf{X}} < \infty$ and $k_{\mathbf{X}} \leq n_{\sigma,\mathbf{X}}$, (ii) $n_{\sigma,\mathbf{X}} < \infty$ and $k_{\mathbf{X}} > n_{\sigma,\mathbf{X}}$, and (iii) $n_{\sigma,\mathbf{X}} = \infty$.

If (i) holds, then $n_{\sigma',\mathbf{X}} = k_{\mathbf{X}}$, and $\pi' = \pi^{\mathsf{f}}[\sigma', \mathbf{X}] = \pi(1) \ldots \pi(k_{\mathbf{X}})$. From the fact that $\pi(1) \ldots \pi(k_{\mathbf{X}})$ is a bad prefix for the safe formula α and the $\mathsf{LTL_f}$ equivalence $\neg\bullet\varphi \equiv \bigcirc\neg\varphi$, we obtain $\pi' \models \neg\alpha'$.

If (ii) holds, then it follows from the definition of σ' that $n_{\sigma',\mathbf{X}} = n_{\sigma,\mathbf{X}}$ and $\pi^{\mathsf{f}}[\sigma', \mathbf{X}] = \pi(1) \ldots \pi(n_{\sigma,\mathbf{X}})$. Since σ is a winning strategy for $\mathcal{S}$ with $n_{\sigma,\mathbf{X}} < \infty$, we must have $\pi(1) \ldots \pi(n_{\sigma,\mathbf{X}}) \models \varphi$, hence $\pi^{\mathsf{f}}[\sigma', \mathbf{X}] \models \varphi$.

If (iii) holds, then it must be the case that $\pi \not\models \alpha$ (because φ is a winning strategy). As α is a safe formula, π must contain a bad prefix for α, so $k_{\mathbf{X}} < \infty$. Using a similar argument as in case (i), we can show that $\pi' \models \neg\alpha'$. □

Theorem 5 The constrained $\mathsf{LTL_f}$ specification $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$ is realizable iff LTL specification $\mathcal{S}^\infty = \langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{alive}, \mathsf{end}\}, \psi_{\alpha,\varphi} \rangle$ is realizable. Moreover, for every winning strategy σ for $\mathcal{S}^\infty$, the strategy σ' defined by $\sigma'(X_1 \cdots X_n) := \sigma(X_1 \cdots X_n) \setminus \{\mathsf{alive}\}$ is a winning strategy for $\mathcal{S}$.

Proof. For the first direction, suppose that $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha, \varphi \rangle$ is realizable, and let σ be a winning strategy for $\mathcal{S}$. By definition, for every $\mathbf{X} \in (2^{\mathcal{X}})^\omega$, the trace $\pi[\sigma, \mathbf{X}]$ satisfies one of the following: (i) it has a finite prefix of length $n_{\sigma,\mathbf{X}} < \infty$ that satisfies φ, or (ii) $n_{\sigma,\mathbf{X}} = \infty$ and $\pi[\sigma, \mathbf{X}] \not\models \alpha$. We define a strategy σ^∞ for $\mathcal{S}^\infty = \langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{alive}, \mathsf{end}\}, \psi_{\alpha,\varphi} \rangle$ as follows:

- $\sigma^\infty(X_1 \cdots X_n) = \sigma(X_1 \cdots X_n) \cup \{\mathsf{alive}\}$, if there is no $k < n$ such that $\mathsf{end} \in \sigma(X_1 \cdots X_k)$
- $\sigma^\infty(X_1 \cdots X_n) = \sigma(X_1 \cdots X_n)$, otherwise

We claim that σ^∞ is a winning strategy for $\mathcal{S}^\infty$. Take some $\mathbf{X} = X_1 X_2 \ldots$, let $\pi = \pi[\sigma, \mathbf{X}]$ and $\pi^\infty = \pi[\sigma^\infty, \mathbf{X}]$. We first show that $\pi^\infty \models \psi_{\mathsf{end}}$. First note that if $\mathsf{end} \in \pi^\infty(i)$, then $\mathsf{end} \in \pi(i)$, which means $\mathsf{end} \notin \pi(i+1)$ (since π contains at most one end). It follows that $\mathsf{alive} \in \pi^\infty(i)$ and $\mathsf{alive} \notin \pi^\infty(i+1)$. Next suppose that $\mathsf{alive} \in \pi^\infty(i)$ but $\mathsf{alive} \notin \pi^\infty(i+1)$. This can only occur if $\mathsf{end} \in \pi(i)$, which implies that $\mathsf{end} \in \pi^\infty(i)$. We have thus shown that $\pi^\infty \models \Box(\mathsf{end} \leftrightarrow \mathsf{alive} \wedge \bigcirc\neg\mathsf{alive})$. We also have $\pi^\infty \models \Box(\mathsf{end} \rightarrow \bigcirc\Box\neg\mathsf{end})$, since $\mathsf{end} \in \pi^\infty(i)$ iff $\mathsf{end} \in \pi(i)$, and π contains at most one occurrence of end.

We next show that $\pi^\infty \models ((\alpha \vee \Diamond\mathsf{end}) \rightarrow \varphi_{\mathsf{inf}})$. First consider the case (i) where π has a finite prefix of length $n_{\sigma,\mathbf{X}} < \infty$ that satisfies φ. Then $\mathsf{end} \in \pi^\infty(n_{\sigma,\mathbf{X}})$, so $\mathsf{alive} \in \pi^\infty(i)$ for $1 \leq i \leq n_{\sigma,\mathbf{X}}$ and $\mathsf{alive} \notin \pi^\infty(i)$ for $i > n_{\sigma,\mathbf{X}}$. It follows that $\pi^\infty \models \varphi_{\mathsf{inf}}$. Next suppose that (ii) holds, i.e. $n_{\sigma,\mathbf{X}} = \infty$ and $\pi \not\models \alpha$. Since α only involves variables from $\mathcal{X} \cup \mathcal{Y}$, and π^∞ coincides with π on $\mathcal{X} \cup \mathcal{Y}$, it follows that $\pi^\infty \not\models \alpha$. As $n_{\sigma,\mathbf{X}} = \infty$, we know that end does not occur in π. The same must hold for π^∞, hence $\pi^\infty \not\models \Diamond\mathsf{end}$. We thus obtain $\pi^\infty \models \neg(\alpha \vee \Diamond\mathsf{end})$, hence $\pi^\infty \models ((\alpha \vee \Diamond\mathsf{end}) \rightarrow \varphi_{\mathsf{inf}})$.

For the other direction, suppose $\mathcal{S}^\infty$ is realizable, and let σ^∞ be a winning strategy for $\mathcal{S}$. We define σ as follows: $\sigma(X_1 \ldots X_n) = \sigma^\infty(X_1 \ldots X_n) \setminus \{\mathsf{alive}\}$. Our aim is to show that σ is a winning strategy for $\mathcal{S}$. Consider some $\mathbf{X} = X_1 X_2 \ldots$, and let $\pi^\infty = \pi[\sigma^\infty, \mathbf{X}]$ and $\pi = \pi[\sigma, \mathbf{X}]$. First note that since $\pi^\infty \models \Box(\mathsf{end} \rightarrow \bigcirc\Box\neg\mathsf{end})$, π^∞ contains at most one occurrence for end, and the same holds for σ. It follows that σ is a valid strategy. Next suppose that $\mathsf{end} \in \pi^\infty(n)$. Then $\pi^{\mathsf{f}}[\sigma, \mathbf{X}] = \pi(1) \ldots \pi(n)$. As $\pi^\infty \models ((\alpha \vee \Diamond\mathsf{end}) \rightarrow \varphi_{\mathsf{inf}})$ and $\pi^\infty \models \Diamond\mathsf{end}$, we have $\pi^\infty \models \varphi_{\mathsf{inf}}$. When combined with $\pi^\infty \models \Box(\mathsf{end} \leftrightarrow \mathsf{alive} \wedge \bigcirc\neg\mathsf{alive})$, we get that $\mathsf{alive} \in \pi^\infty(i)$ for $i \leq n$ and $\mathsf{alive} \notin \pi^\infty(i)$ for $i > n$. This, together with the definition of φ_{inf}, implies that the induced finite trace $\pi(1) \ldots \pi(n)$ satisfies φ. Finally, consider the case where end does not occur in π^∞. We know that $\pi^\infty \models ((\alpha \vee \Diamond\mathsf{end}) \rightarrow \varphi_{\mathsf{inf}})$. Note that we cannot have $\pi^\infty \models \varphi_{\mathsf{inf}}$ since it implies, when combined with $\pi^\infty \models \Box(\mathsf{end} \leftrightarrow \mathsf{alive} \wedge \bigcirc\neg\mathsf{alive})$, that end occurs in π^∞. It follows that $\pi^\infty \models \neg(\alpha \vee \Diamond\mathsf{end})$, or equivalently, $\pi^\infty \models \neg\alpha \wedge \Box\neg\mathsf{end}$. From this, we can derive that $\pi \not\models \alpha$ and π does not contain end. We have thus established that σ is a winning strategy. □

Theorem 7 The DBA $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ accepts infinite traces π such that either: (i) $\pi(1) \cdots \pi(n) \models \varphi$ and end occurs only in $\pi(n)$, or (ii) $\pi \not\models \alpha_s \wedge \alpha_c$ and end does not occur in π. $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ can be constructed in double-exponential time in $|\alpha_s| + |\alpha_c| + |\varphi|$.

Proof. First, we prove that the language of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ contains the set of traces that satisfy one of the conditions (i) and (ii). Then, we prove that the language of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ is contained in the set of traces that satisfy one of the conditions (i) and (ii). The construction of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ is polynomial in the size of $\mathcal{A}_s$, $\mathcal{A}_c$, and $\mathcal{A}_g$, which can be constructed in double-exponential time in $|\alpha_s|$, $|\alpha_c|$, and $|\alpha_g|$, respectively.

($\supseteq$) (i) If $\pi(1)\cdots\pi(n) \models \varphi$ and end occurs only in $\pi(n)$, then the run of $\mathcal{A}_g$ on $\pi(1)\cdots\pi(n)$ is accepting, i.e. its last state belongs to F_g. It then follows from the definition of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ that the run of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ on π will transition to $q_\top$ after reading $\pi(n)$ and, because end occurs only in $\pi(n)$, it will then loop at $q_\top$. As $q_\top$ is an accepting state of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$, this shows that $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ accepts π. (ii) If $\pi \not\models \alpha_s \wedge \alpha_c$ and end does not occur in π, then either $\pi \not\models \alpha_s$ or $\pi \not\models \alpha_c$. In the first case ($\pi \not\models \alpha_s$), the trace π contains a bad prefix $\pi(1)\cdots\pi(n)$, and w.l.o.g. we can suppose that this is the shortest such prefix. It follows that the run of $\mathcal{A}_s$ on $\pi(1)\cdots\pi(n)$ is accepting. From the definition of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$, the run of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ on π transitions to a state (q_s, q_c, q_g) where $q_s = q_{\mathsf{bad}}$. Because end does not occur in π, after having read $\pi(n)$, the run of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ on π will remain among the states whose third component is q_{bad}. By construction, these states are accepting, so $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ accepts π. In the second case ($\pi \not\models \alpha_g$), the run of $\mathcal{A}_c$ on finite prefix $\pi(1)\cdots\pi(n)$ is not accepting for any $n < \infty$. It follows that the run of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ on π will not visit any state (q_s, q_c, q_g) with $q_c = q_{\mathsf{good}}$, and since π does not contain end, it also cannot contain the state $q_\perp$. Thus, the run of π will only visit accepting states, so $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ accepts π.

($\subseteq$) Let π be an infinite trace that is accepted by $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$. We distinguish three cases: (a) end does not occur in π; (b) end occurs exactly one time in π; (c) end occurs more than one time in π. Case (a): if end does not occur in π, then $q_\top$ does not occur in the run of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ on π. As this run is accepting but does not contain $q_\top$, it must either hit infinitely often states with q_{bad}, or hit infinitely often states without q_{good}. In the first case, let $\pi(1)\cdots\pi(n)$ be the smallest prefix of π such that after reading $\pi(1)\cdots\pi(n)$, the DBA $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ enters a state (q_s, q_c, q_g) with $q_s = q_{\mathsf{bad}}$. By construction of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$, it must be that the run of $\mathcal{A}_s$ on $\pi(1)\cdots\pi(n)$ is accepting. Thus, $\pi(1)\cdots\pi(n)$ is a bad prefix of α_s, which means $\pi \not\models \alpha_s$. In the second case, we observe that if a run enters a state (q_s, q_c, q_g) with $q_c = q_{\mathsf{good}}$, then it remains in a state with good in the second component unless a symbol with end is read. As the considered trace π does not contain end, it follows that the run of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ on π does not hit any state (q_s, q_c, q_g) with $q_c = q_{\mathsf{good}}$. Hence, $\mathcal{A}_c$ must not accept any finite prefix of π, so π does not contain any good prefixes for α_c, i.e. $\pi \not\models \alpha_c$. This concludes that case (a) implies case (ii). Case (b): suppose that end occurs exactly one time in π, say in the nth symbol $\pi(n)$. Immediately after reading $\pi(n)$, the DBA $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ will transition to either $q_\top$ or $q_\perp$. However, since the run of $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ on π is accepting, it cannot contain $q_\perp$. We thus have a transition of the form $\delta((q_s, q_c, q_g), \pi(n)) = q_\top$ with $\delta_g(q_g, \pi(n)) \in F_g$. It follows that the finite prefix $\pi(1)\cdots\pi(n)$ is accepted by the DFA $\mathcal{A}_g$, i.e., $\pi(1)\cdots\pi(n) \models \varphi$. We have thus shown that case (b) implies case (i). To conclude the proof, observe that Case (c) is not possible. Indeed, the first symbol with end forces an automaton transition to either $q_\top$ or $q_\perp$, and the second symbol with end forces an automaton transition to $q_\perp$. Because state $q_\perp$ is absorbing and not accepting, $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ does not accept traces where end occurs more than one time. □

Theorem 9 Consider a constrained $\mathsf{LTL_f}$ specification $\mathcal{S} = \langle \mathcal{X}, \mathcal{Y}, \alpha_s \wedge \alpha_c, \varphi \rangle$ where α_s (resp. α_c) is a safe (resp. co-safe) formula. Then:

- $\mathcal{S}$ is realizable iff the DBA game $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ is winning;
- Every winning strategy for $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ is a winning strategy for $\mathcal{S}$, and vice-versa;
- Deciding whether $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ is winning, and constructing a winning strategy when one exists, can be done in 2EXP.

Proof. By Theorem 7, the language of the DBA $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ contains all, and only, the infinite traces π such that either: (i) $\pi(1)\cdots\pi(n) \models \varphi$ and end occurs only in $\pi(n)$, or (ii) $\pi \not\models \alpha_s \wedge \alpha_c$ and end does not occur in π. By definition, a strategy σ is a winning strategy for $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ iff every induced infinite trace $\pi[\sigma, \mathbf{X}]$ is accepted by $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$. It follows that winning strategies for $\mathcal{G}_\varphi^{\alpha_s,\alpha_c}$ are winning strategies for $\mathcal{S}$, and vice versa. By Theorem 7, we can construct the DBA $\mathcal{A}_\varphi^{\alpha_s,\alpha_c}$ in double-exponential time. We can then construct, in polynomial time in $|\mathcal{A}_\varphi^{\alpha_s,\alpha_c}|$ the set $Win(\mathcal{G}_\varphi^{\alpha_s,\alpha_c})$ and check whether $((q_0)_s, (q_0)_c, (q_0)_g) \in Win(\mathcal{G}_\varphi^{\alpha_s,\alpha_c})$. If the latter holds, then we construct a winning strategy in the form of a transducer. This step is also polynomial w.r.t. $|\mathcal{A}_\varphi^{\alpha_s,\alpha_c}|$, and thus the entire procedure can be performed in double-exponential time. □

Theorem 10 A bgv-optimal strategy can be constructed in double-exponential time.

Proof. We first argue that the described construction yields an bgv-optimal strategy. We know from (Almagor, Boker, and Kupferman 2016) that the $\hat{\mathcal{A}}_{\varphi,\mathcal{V}}$ accepts finite traces π such that $[\![\pi, \varphi]\!] \geq b$. By reusing the arguments from the proof of Theorem 7, we can show that $\mathcal{A}^\alpha_{\varphi \geq b}$ recognizes infinite traces such that either (i) $\alpha = \alpha_s \wedge \alpha_c$ is violated and end does not occur, or (ii) end occurs exactly once and the induced finite trace π is such that $[\![\pi, \varphi]\!] \geq b$. It follows that an α-strategy σ with $\mathsf{bgv}(\sigma) \geq b$ exists iff the DBA game $\langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{end}\}, \mathcal{A}^\alpha_{\varphi \geq b} \rangle$ is winning. We can use the techniques described in Section 5.2 to decide whether such a DBA is winning. By considering the values in $V(\varphi)$ in descending order, we can determine the maximal b^* for which an α-strategy σ with $\mathsf{bgv}(\sigma) \geq b^*$ exists. We can then compute such a strategy, for the identified best value b^*, by constructing a winning strategy for the DBA game $\langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{end}\}, \mathcal{A}^\alpha_{\varphi \geq b^*} \rangle$, again using the techniques from Section 5.2.

For each value b, we can construct the DFA $\hat{\mathcal{A}}_{\varphi,\mathcal{V}}$ in double-exponential time, and the DFAs for safe and co-safe constraints can also be constructed in double-exponential time. It follows that $\mathcal{A}^{\alpha}_{\varphi\geq b}$ can also be constructed in double-exponential time. Determining whether the DBA game $\langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{end}\}, \mathcal{A}^{\alpha}_{\varphi\geq b}\rangle$ is winning, and constructing a winning strategy for the game, is also in double-exponential time. Finally, we note that all of the preceding double-exponentional time operations are performed at most once per value $b \in V(\varphi)$, so the overall procedure runs in double-exponential time. □

The next two lemmas will be used to prove Theorem 11. In what follows, it will be convenient to slightly abuse notation and use $\delta(q, \pi)$, with π a finite trace, to indicate the automata state resulting from reading π starting from state q (and similarly for the output function ω of transducers on a finite string $X_1 \dots X_n$).

Lemma 1. *For every $\pi = (X_1 \cup Y_1) \dots (X_n \cup Y_n) \in 2^{\mathcal{X}\cup\mathcal{Y}}$, $X_{n+1} \in 2^{\mathcal{X}}$, and $b \in [0, 1]$, the following are equivalent:*

1. *there exists an α-strategy σ such that $\pi \in \mathsf{ptraces}(\sigma)$ and $\mathsf{bgv}_{\pi,X_{n+1}}(\sigma) \geq b$*
2. *$\delta_b(q^b_0, \pi\cdot(X_{n+1}\cup Y_{n+1})) \in Win(\mathcal{G}_b)$ for some $Y_{n+1} \in 2^{\mathcal{Y}\cup\{\mathsf{end}\}}$.*

Proof. (⇒) Suppose that σ is an α-strategy such that $\pi \in \mathsf{ptraces}(\sigma)$ and $\mathsf{bgv}_{\pi,X_{n+1}}(\sigma) \geq b$. Set $Y_{n+1} = \sigma(X_1 \dots X_n X_{n+1})$, and let $q^* = \delta_b(q^b_0, \pi\,(X_{n+1} \cup Y_{n+1}))$. Consider the DBA $\mathcal{A}^{\alpha}_{\geq b,q^*} = \langle 2^{\mathcal{X}\cup\mathcal{Y}\cup\{\mathsf{end}\}}, Q_b, \delta_b, q^*, F_b\rangle$, which is the same as $\mathcal{A}^{\alpha}_{\geq b}$ but with q^* for the initial state. Define a strategy σ^* for the DBA game $\mathcal{G}^*_b = \langle \mathcal{X}, \mathcal{Y} \cup \{\mathsf{end}\}, \mathcal{A}^{\alpha}_{\geq b,q^*}\rangle$ as follows:

$$\sigma^*(X'_1 \dots X'_h) = \sigma(X_1 \dots X_{n+1} X'_1 \dots X'_h)$$

We claim that σ^* is a winning strategy for $\mathcal{G}^*_b$. To see why, take any $\mathbf{X} \in (2^{\mathcal{X}})^*$, and let $\pi^*_{\mathbf{X}} = \pi[\sigma^*, \mathbf{X}]$. We need to show that $\pi^*_{\mathbf{X}}$ is accepted by $\mathcal{A}^{\alpha}_{\geq b,q^*}$. Let us consider $\pi_{\mathbf{X}} = \pi[\sigma, X_1 \dots X_{n+1}\mathbf{X}]$, and let $s_0 s_1 s_2 \dots$ be the infinite run of $\mathcal{A}^{\alpha}_{\geq b}$ on $\pi_{\mathbf{X}}$. Note that since $\pi \in \mathsf{ptraces}(\sigma)$, we know that $\pi \sqsubseteq \pi_{\mathbf{X}}$, hence either (i) $\pi_{\mathbf{X}}$ does not contain end and $\pi_{\mathbf{X}} \not\models \alpha$, or (ii) $\pi_{\mathbf{X}}$ contains exactly one occurrence of end (at some position $k \leq n + 1$, since π does not contain end) and the induced finite trace $\pi^{\mathsf{f}}_{\mathbf{X}} = \pi^{\mathsf{f}}[\sigma, X_1 \dots X_{n+1}\mathbf{X}]$ is such that $[\![\pi^{\mathsf{f}}_{\mathbf{X}}, \varphi]\!] \geq b$. It follows that $\pi_{\mathbf{X}}$ is accepted by $\mathcal{A}^{\alpha}_{\geq b}$, and thus the infinite run $s_0 s_1 s_2 \dots$ of $\mathcal{A}^{\alpha}_{\geq b}$ on $\pi_{\mathbf{X}}$ contains infinitely many $s_j \in F_b$. We then observe that $s_{n+1} = q^*$. Since $\mathcal{A}^{\alpha}_{\geq b,q^*}$ has the same transitions as $\mathcal{A}^{\alpha}_{\geq b}$, it follows that $s_{n+1} s_{n+2} s_{n+3} \dots$ is the run of $\mathcal{A}^{\alpha}_{\geq b,q^*}$ on $\pi^*_{\mathbf{X}}$. Note that $s_{n+1} s_{n+2} s_{n+3} \dots$ must also contain infinitely many $s_j \in F_b$, so it is an accepting run. We have thus shown that σ^* is a winning strategy for $\mathcal{G}^*_b$.

As the game $\mathcal{G}^*_b$ is winning, we must have $q^* \in Win(\mathcal{G}^*_b)$. We then remark that since $\mathcal{G}^*_b$ and $\mathcal{G}_b$ only differ in their initial states, the two games have precisely the same winning regions. We thus obtain $q^* = \delta_b(q^b_0, \pi\,(X_{n+1} \cup Y_{n+1})) \in Win(\mathcal{G}_b)$.

(⇐) Suppose that $q^* = \delta_b(q^b_0, \pi\cdot(X_{n+1} \cup Y_{n+1})) \in Win(\mathcal{G}_b)$. By following the strategy of the transducer $\mathcal{T}_{\mathcal{G}_b}$ from this point on, we are guaranteed to produce a trace that is accepted by the automaton $\mathcal{A}^{\alpha}_{\geq b}$. More precisely, consider the strategy σ defined as follows:

- $\sigma(X'_1 \dots X'_k) = Y_k$, if $X'_1 \dots X'_k = X_1 \dots X_k$ $(1 \leq k \leq n + 1)$
- $\sigma(X'_1 \dots X'_{n+1} \dots X'_h) = \omega_b(\delta'_b(q^*, X'_{n+1} \dots X'_{h-1}), X'_h)$, if $X'_1 \dots X'_{n+1} = X_1 \dots X_{n+1}$
- $\sigma(X'_1) = \mathsf{end}$, if $X'_1 \neq X_1$
- $\sigma(X'_1 \dots X'_k X'_{k+1}) = \mathsf{end}$, if $X'_i = X_i$ for $1 \leq k < n + 1$ and $X'_{k+1} \neq X_{k+1}$
- $\sigma(X'_1 \dots X'_h) = \emptyset$, in all other cases

where δ'_b and ω_b are respectively the transition and output functions of the transducer $\mathcal{T}_{\mathcal{G}_b}$. The first bullet concerns prefixes of $X_1 \dots X_{n+1}$ and ensures that $(X_1 \cup Y_1) \dots (X_{n+1} \cup Y_{n+1}) \in \mathsf{ptraces}(\sigma)$). The second bullet states that once $X_1 \dots X_{n+1}$ has been read, we start following the transducer $\mathcal{T}_{\mathcal{G}_b}$. The remaining points ensure that all infinite traces $\pi' \in \mathsf{traces}(\sigma)$ such that $\pi\,(X_{n+1} \cup Y_{n+1}) \not\sqsubseteq \pi'$ contain a single occurrence of end.

We claim that σ is an α-strategy such that $(X_1 \cup Y_1) \dots (X_{n+1} \cup Y_{n+1}) \in \mathsf{ptraces}(\sigma)$ and $\mathsf{bgv}_{\pi,X_{n+1}}(\sigma) \geq b$. As noted above, the first bullet of the definition ensures that $(X_1 \cup Y_1) \dots (X_{n+1} \cup Y_{n+1}) \in \mathsf{ptraces}(\sigma)$. The last three bullets make sure that every $\pi' \in \mathsf{traces}(\sigma)$ with $(X_1 \cup Y_1) \dots (X_{n+1} \cup Y_{n+1}) \not\sqsubseteq \pi'$ contains exactly one occurrence of end. It remains to consider the infinite traces that begin with $(X_1 \cup Y_1) \dots (X_{n+1} \cup Y_{n+1})$. Consider some such trace $\pi' = (X_1 \cup Y_1)(X_2 \cup Y_2) \dots \in \mathsf{traces}(\sigma)$, and let $s_0 s_1 s_2 \dots$ be the infinite run of $\mathcal{A}^{\alpha}_{\geq b}$ on π'. Because $(X_1 \cup Y_1) \dots (X_{n+1} \cup Y_{n+1}) \sqsubseteq \pi'$, we know that $s_{n+1} = q^*$. The latter can be combined with the second bullet to show that for every $i \geq n+1$, $Y_{i+1} = \omega_b(s_i, X_{i+1})$ and $s_{i+1} = \delta'_b(s_i, X_{i+1})$. As $s_{n+1} = q^* \in Win(\mathcal{G}_b)$, and ω_b is defined so as to always remain within $Win(\mathcal{G}_b)$, we have $s_i \in Win(\mathcal{G}_b)$ for every $i \geq n + 1$. Furthermore, since ω_b always reduces the distance to $RA(\mathcal{G})$, there are infinitely many $i \geq n + 1$ such that $s_i \in F_b$. This establishes that $s_0 s_1 s_2 \dots$ is an accepting run, and thus, π' must either be such that $\pi' \not\models \alpha$, or it contains a single occurrence of end such that the finite induced trace π'' is such that $[\![\pi'', \varphi]\!] \geq b$. We have thus shown that every infinite trace produced by σ either violates α or contains end, i.e. σ is an α-strategy. Moreover, for every $\pi'' \in \mathsf{traces}^{\mathsf{f}}(\sigma)$ with $(X_1 \cup Y_1) \dots (X_{n+1} \cup Y_{n+1}) \sqsubseteq \pi''$, we have $[\![\pi'', \varphi]\!] \geq b$, as desired. □

In the following lemma, we use σ_{str} to denote the strategy implemented by the transducer $\mathcal{T}^{\mathsf{str}}$. To simplify the formulation, we extend the notation $\mathsf{bgv}_{\pi,X}(\sigma)$ to all finite traces that can be produced by strategy σ (recall that Definition 2 only defines this notation for compatible traces that do not contain end). Formally, given a trace $\pi = (X_1 \cup Y_1) \dots (X_n \cup Y_n)$ that is a prefix of some infinite trace in $\pi[\sigma, \mathbf{X}]$ and contains end at position $k \leq n$, we set $\mathsf{bgv}_{\pi,X}(\sigma) = [\![\pi', \varphi]\!]$, where $\pi' = (X_1 \cup Y_1) \dots (X_k \cup Y_k \setminus \{\mathsf{end}\})$.

Lemma 2. *For every $X_1 \dots X_n \in (2^{\mathcal{X}})^*$, $X_{n+1} \subseteq \mathcal{X}$, and $v \in [0, 1]$: if $\pi = (X_1 \cup \omega_{\mathsf{str}}(q^{\mathsf{str}}_0, X_1)) \dots (X_n \cup \omega_{\mathsf{str}}(q^{\mathsf{str}}_0, X_1 \dots X_n))$, $(q_1, q_2, \dots, q_m) = \delta_{\mathsf{str}}(q^{\mathsf{str}}_0, X_1 \dots X_n)$, and u is such that*

$\omega_{\mathsf{str}}((q_1, q_2, \ldots, q_m), X_{n+1})$ *was set equal to* $\omega_u(q_u, X_{n+1})$, *then* $\mathsf{bgv}_{\pi,X_{n+1}}(\sigma_{\mathsf{str}}) \geq u$.

Proof. Fix $X_1 \ldots X_n \in (2^{\mathcal{X}})^*$ and $X_{n+1} \subseteq \mathcal{X}$. Consider some $\mathbf{X} \in (2^{\mathcal{X}})^\omega$ such that $\mathbf{X} = X_1 \ldots X_n X_{n+1} X_{n+2} \ldots$. Let $\pi' = \pi[\sigma_{\mathsf{str}}, \mathbf{X}] = (X_1 \cup Y_1)(X_2 \cup Y_2) \ldots$, and for all $i \geq 0$ let $(q^1_{i+1}, \ldots, q^m_{i+1}) = \delta_{\mathsf{str}}(q_0^{\mathsf{str}}, X_1 \ldots X_{i+1})$. Define a function $\zeta : \mathbb{N} \to \{b_1, \ldots, b_m\}$ by letting $\zeta(i)$ be the unique value b in $\{b_1, \ldots, b_m\}$ such that $\omega_{\mathsf{str}}((q^1_i, \ldots, q^m_i), X_1 \ldots X_i), X_{i+1})$ was set equal to $\omega_b(q^b_i, X_{i+1})$. Basically, $\zeta(i) = b$ means that we used transducer $\mathcal{T}_{\mathcal{G}_b}$ to generate Y_{i+1} after reading X_{i+1} from state $(q^1_i, \ldots, q^m_i)$. Note that the value u from the lemma statement is equal to $\zeta(n)$.

Claim: For all $i \geq 0$, $\zeta(i+1) \geq \zeta(i)$.
Proof of claim: Suppose that $\zeta(i) = b$, which means that $b = \mathsf{max}(\{v \in B \mid \exists Y\, \delta_v(q^v_i, X_{i+1} \cup Y) \in Win(\mathcal{G}_v)\})$. Then we have $Y_{i+1} = \omega_b(q^b_i, X_{i+1})$. From the definition of ω_b, and the fact that there exists some Y with $\delta_b(q^b_i, X_{i+1} \cup Y) \in Win(\mathcal{G}_b)$, we know that $\delta_b(q^b_i, X_{i+1} \cup Y_{i+1}) \in Win(\mathcal{G}_b)$. It follows that there must exist Y' such that $\delta_b(q^b_{i+1}, X_{i+2} \cup Y') \in Win(\mathcal{G}_b)$, and hence $\mathsf{max}(\{v \in B \mid \exists Y'\, \delta_v(q^v_{i+1}, X_{i+2} \cup Y') \in Win(\mathcal{G}_v)\}) \geq b$. Thus, $\zeta(i+1) \geq b = \zeta(i)$. (*end proof of claim*)

Due to the preceding claim, and the finiteness of B, there exists $h \geq 0$ such that $\zeta(i) = \zeta(h)$ for all $i \geq h$, and $\zeta(i) < \zeta(h)$ for $i < h$. Let $b_{\mathbf{X}} = \zeta(h)$. Observe that $q_0^{b_{\mathbf{X}}} q_1^{b_{\mathbf{X}}} q_3^{b_{\mathbf{X}}} \ldots$ is the run of $\mathcal{A}^\alpha_{\geq b_{\mathbf{X}}}$ on π' and that $q^{b_{\mathbf{X}}}_{h+1} \in Win(\mathcal{G}_{b_{\mathbf{X}}})$. From the definition of $\omega_{b_{\mathbf{X}}}$, we can infer that $q_i^{b_{\mathbf{X}}} \in Win(\mathcal{G}_{b_{\mathbf{X}}})$ for all $i > h$, and further that there are infinitely many $q_i^{b_{\mathbf{X}}} \in F_{b_{\mathbf{X}}}$. This means that $q_0^{b_{\mathbf{X}}} q_1^{b_{\mathbf{X}}} q_3^{b_{\mathbf{X}}} \ldots$ is an accepting run of $\mathcal{A}^\alpha_{\geq b_{\mathbf{X}}}$ on π'. Thus, either π' does not contain end and violates α, or π' contains end and the induced finite trace $\pi^{\mathsf{f}}[\sigma_{\mathsf{str}}, \mathbf{X}]$ has value at least $b_{\mathbf{X}} = \zeta(h) \geq \zeta(n) = u$. We have shown that every trace $\pi'' \in \mathsf{traces}^{\mathsf{f}}(\sigma_{\mathsf{str}})$ is such that $[\![\pi'', \varphi]\!] \geq \zeta(n) = u$, and hence that $\mathsf{bgv}_{\pi,X_{n+1}}(\sigma_{\mathsf{str}}) \geq u$. □

We now proceed to the proof of Theorem 11.

Theorem 11 $\mathcal{T}^{\mathsf{str}}$ implements a strongly bgv-optimal strategy and can be constructed in double-exponential time.

Proof. We first show that $\mathcal{T}^{\mathsf{str}}$ implements a strongly bgv-optimal strategy. Suppose for a contradiction that this is not the case. Then there exists a finite trace $\pi = (X_1 \cup Y_1) \ldots (X_n \cup Y_n) \in \mathsf{ptraces}(\sigma)$ that does not contain end, $X \in \mathbf{X}$, and an α-strategy such that $\pi \in \mathsf{ptraces}(\sigma')$ and $\mathsf{bgv}_{\pi,X}(\sigma') > \mathsf{bgv}_{\pi,X}(\sigma)$. Suppose that $\mathsf{bgv}_{\pi,X}(\sigma) = b$ and $\mathsf{bgv}_{\pi,X}(\sigma') = b'$. From Lemma 1, we know that $\delta_{b'}(q_0^{b'}, \pi\,(X_{n+1} \cup Y_{n+1})) \in Win(\mathcal{G}_{b'})$ for some $Y_{n+1} \in 2^{\mathcal{Y} \cup \{\mathsf{end}\}}$. It follows that $\omega_{\mathsf{str}}(\delta_{\mathsf{str}}(q_0^{\mathsf{str}}, X_1 \ldots X_n), X_{n+1}) = \omega_v(q_v, X_{n+1})$ for some $v \geq b'$. From Lemma 2, $\mathsf{bgv}_{\pi,X_{n+1}}(\sigma_{\mathsf{str}}) \geq v \geq b'$, a contradiction.

We know from Section 7.2 and Theorem 10 that for each $v \in V(\varphi)$, the DBA $\mathcal{A}^\alpha_{\geq v}$, winning region $Win(\mathcal{G}_v)$, and transducer $\mathcal{T}_{\mathcal{G}_v}$ can be constructed in double-exponential time. As there are only single exponentially many values in V (hence B), the transducer $\mathcal{T}^{\mathsf{str}}$ can also be constructed double-exponential time. □